Universidade de Aveiro
2025

Afonso Ferreira
Carolina Silva
Pedro Ponte
Ricardo Antunes
Tomás Brás

EvalMed
Reliable Professional Activity Management System for Medical Training in UA

Universidade de Aveiro
2025

Afonso Ferreira
Carolina Silva
Pedro Ponte
Ricardo Antunes
Tomás Brás

EvalMed presented at the University of Aveiro in fulfillment of the requirements for the completion of the curricular unit Projeto de Informática, a necessary condition for obtaining the degree of Bachelorette in Computer Science , realized with the orientation of Carlos Costa, Professor associate with habilitation of Departamento de Eletrónica, Telecomunicações e Informática in Universidade of Aveiro and co-orientation of José Luís Oliveira full professor at the Departamento de Eletrónica, Telecomunicações e Informática of Universidade de Aveiro.

agradecimentos / acknowledgements

We would like to express our deepest gratitude to our supervisor, Professor Carlos Costa, for all the guidance and knowledge shared throughout the semester and for always being available to help us. We also would like to thank Professors João Firmino Machado, Sílvia Bidarra and José Pedro Antunes for their enthusiasm in making the project come to life by providing constant feedback about system requirements and being the bridge between the team and the end users. Lastly, a special thanks to professor João Almeida for putting up with our weekly requests and finally everyone who in one way or another contributed to the project, especially our usability tests participants.

Keywords Trusted Professional Activity Management, EPA, Communication between Student and Tutor, MEC, Medicine

Abstract

EvalMed is a digital solution developed to upgrade the management of EPA in medical education at the University of Aveiro. The system addresses the limitations of traditional pen-and-paper based organization methods by providing a platform for students, tutors, and administrators to manage practical evaluations effectively. The application offers several key functionalities including user authentication via the university's identity provider, real-time feedback mechanisms, performance analytics, and offline capabilities. EvalMed streamlines the evaluation process by allowing tutors to assess students' practical skills through a structured framework while enabling students to track their progress, request reevaluations, and access educational materials specific to each MEC. The system directly supports the CAM framework, which follows the seven CanMeds model medical competencies and establishes clear autonomy levels (N1–N3) for each clinical activity. Through EvalMed, tutors can document student progress across a wide range of MECs during the scheduled clinical residency sessions at USF, providing structured feedback at designated milestones. EvalMed represents a significant advancement in practical medical education evaluation, replacing paper-based methods with a responsive, user-friendly digital solution that enhances feedback quality and simplifies performance tracking throughout the academic year.

Contents

Contents i

List of Figures v

List of Tables vii

List of Code Snippets ix

Glossary xi

1 Introduction 1

1.1 Context and Motivation 1

1.2 Objectives 2

1.3 Structure of the Document 2

2 State of the Art 3

2.1 Current Paper Solution for the Entrustable Professional Activity (EPA) Methodology 3

2.2 Existing Software Solutions for the EPA Methodology 4

2.2.1 Major Features of Existing Solutions 5

2.2.2 Limitations of Existing Solutions 5

3 Requirements of the System and the Architecture 7

3.1 Requirements of the System 7

3.1.1 Personas, Scenarios & User Stories 8

3.1.1.1 Student User Stories 12

3.1.1.2 Tutor User Stories 15

3.1.1.3 Administrator User Stories 17

3.1.2 Requirement Elicitation 21

3.1.2.1 Functional Requirements 21

3.1.2.2 Non-Functional Requirements 24

3.2 Architecture 26

3.2.1 Backend 26
3.2.2 Databases 27
3.2.2.1 MySQL 27
3.2.2.2 MongoDB 27
3.2.3 Observability 28
3.2.4 Frontend 28
3.2.5 Middleware 28
3.2.6 Containerization 28
3.2.7 Domain model 29
3.2.7.1 MySQL 29
3.2.7.2 MongoDB 30
3.2.8 Architecture Diagram 33
3.3 Deployment Diagram 34
3.3.1 Docker 34
3.3.2 Docker Compose 35
3.3.3 Production Virtual Machine (VM) 35

4 Implementation 37
4.1 Web Application 37
4.1.1 Architecture 37
4.1.2 Organization 37
4.1.3 Frontend 38
4.1.3.1 Frontend Folder Structure 38
4.1.3.2 Configuration Files 38
4.1.3.3 Service Workers 40
4.1.3.4 Prototype 42
4.1.3.5 Responsiveness and Cross-Device Compatibility 43
4.1.3.6 Implemented Main Functionalities 44
4.1.3.7 Student 44
4.1.3.8 Tutor 52
4.1.3.9 Administrator 56
4.1.3.10 Core Functionalities and Technical Implementation 63
4.1.3.11 Support Page 63
4.1.4 Backend 64
4.1.4.1 Folder Structure 64
4.1.5 Nginx Configuration 66
4.1.6 Backups 66
4.1.7 API 66

4.2 Notification System . . . 67
4.2.1 Push Notifications . . . 67
4.2.2 Email Notifications . . . 68
4.3 Authentication and Authorization . . . 69
4.3.1 Implemented Authentication Methods . . . 69
4.3.2 Authentication Flows Overview . . . 69
4.3.3 User Access Control . . . 70
4.3.4 Token Management and Session Handling . . . 70
4.4 Security Considerations . . . 72
4.4.1 Infrastructural Security . . . 72
4.4.2 Software Security . . . 72
4.4.2.1 Reverse Proxy and Rate Limiting . . . 72
4.4.2.2 Authenticated Application Programming Interface (API) Requests 72
4.4.2.3 Role Based Access Control (RBAC) . . . 72
4.4.2.4 Frontend Security Measures . . . 73
4.4.3 Data Security . . . 73
4.4.3.1 Authentication . . . 73
4.4.3.2 General Data Protection Regulation (GDPR) . . . 74
4.5 Observability and Monitoring . . . 75
4.6 CI/CD Workflows . . . 76

5 Tests and Results **79**
5.1 Introduction . . . 79
5.2 Usability Testing . . . 79
5.2.1 Low-Fidelity Prototype Testing . . . 80
5.2.1.1 Sample . . . 80
5.2.1.2 Method . . . 80
5.2.1.3 Results . . . 81
5.2.2 High-Fidelity Prototype Testing . . . 86
5.2.2.1 Sample . . . 86
5.2.2.2 Method . . . 86
5.2.2.3 Results . . . 87
5.3 Load Testing . . . 91
5.4 Conclusion of Testing . . . 91

6 Conclusion **93**
6.1 Project Overview . . . 93
6.2 Main Results . . . 93

6.3 Future Work 94

References 95

A Appendix A - API Documentation 95

B Appendix B - Student Task Form (Low Fidelity Prototype) 143

C Appendix C - Tutor Task Form (Low Fidelity Prototype) 149

D Appendix D - Observer Form (Low Fidelity Prototype) 155

E Appendix E - Post-Task Questionnaire (Low Fidelity Prototype) 167

F Appendix F - Student Task Form (High Fidelity Prototype) 173

G Appendix G - Observer Form (High Fidelity Prototype) 181

H Appendix H - SUS Questionnaire (Final Testing) 193

I Appendix I - Personal Data Consent Form 199

J Appendix J - Image Consent Form 201

List of Figures

2.1 Example of a Medical Activities Notebook (CAM) used in the Medicine Course at University of Aveiro. 4

3.1 Use Case Diagram 21
3.2 Entity-Relationship Diagram (ERD) (Entity Relationship Diagram) 31
3.3 Entity-Relationship (ER) (Entity-Relationship) 32
3.4 Architecture Diagram 33
3.5 Deployment Diagram 34

4.1 Folder structure 38
4.2 Folder structure of the application's frontend 39
4.3 Responsive view of the student home page on a mobile device 43
4.4 Student's main interface as displayed on a desktop device 44
4.5 Page of a specific evaluation where the student completes the self-assessment 45
4.6 Student – Feedback submission interface 46
4.7 Student – Reevaluation request form 47
4.8 Student – List of all MECs 48
4.9 Student – Filtering MECs by 'Approval Status' and 'Current Autonomy Level' 48
4.10 Student – Procedure view page 49
4.11 Student – Evaluations list 50
4.12 Student – Location feature 51
4.13 Student – Past curricular year details (Profile Information) 52
4.14 Student – Past curricular year details (MECs and Evaluations) 52
4.15 Tutor – Changing between students 53
4.16 Tutor – Performing an evaluation 53
4.17 Tutor – Feedback submission and response interface 54
4.18 Tutor – Reevaluation response screen 55
4.19 Tutor – Evaluation planning interface 56
4.20 Administrator – Year Selector 57
4.21 Administrator – Global statistics dashboard 57

4.22 Administrator – Student profile view 58
4.23 Administrator – Manage evaluations 59
4.24 Administrator – User management interface 60
4.25 Administrator – MEC procedures management 61
4.26 Example of the Tutor's support page 64
4.27 Folder structure of the application's backend 64
4.28 Notification header with the number of unread notifications 67
4.29 Email notification example 68
4.30 Grafana Dashboard for Monitoring Backend Services 76

5.1 Perceived difficulty ratings for student tasks (best, average, worst). 82
5.2 Perceived difficulty ratings for tutor tasks (best, average, worst). 83
5.3 Tasks completed with correct results during usability tests. 84
5.4 Median score per question on the System Usability Scale (SUS). 85
5.5 Perceived difficulty ratings for student tasks (best, average, worst). 88
5.6 Tasks completed with correct results during usability tests. 90
5.7 Median scores per question on the SUS. 90
5.8 Summary of API performance during the load test using K6. 91
5.9 Average score per question in the SUS questionnaire. 92

List of Tables

4.1 Platform Functionalities by User Role 62

5.1 Average time (min:sec) and errors per task (n = 5) 84
5.2 Average perceived difficulty per task (1 = Very easy, 5 = Very difficult) 84
5.3 Average time (min:sec) and average errors per task (n=4 participants) 89
5.4 Average observed difficulty per task (1 = Very easy, 5 = Very difficult) 89
5.5 Qualitative summary per task 89

List of Code Snippets

4.1 Firebase Service Worker 40

Glossary

sTIC	Information and Communication Technologies services
EPA	Entrustable Professional Activity
MEC	Micro Entrustable Competency
USF	Family Health Unit
CAM	Medical Activities Notebook
OTT	One-Time Token
IdP	Identity Provider
PWA	Progressive Web Application
OIDC	OpenID Connect
JWT	JSON Web Token
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
JS	JavaScript
API	Application Programming Interface
IEETA	Institute of Electronics and Informatics Engineering of Aveiro
UA	University of Aveiro
VM	Virtual Machine
ERD	Entity-Relationship Diagram
ER	Entity-Relationship
SUS	System Usability Scale
GDPR	General Data Protection Regulation
DB	Database
SQL	Structured Query Language
FCM	Firebase Cloud Messaging
SW	Service Worker
ESSUA	Escola Superior de Saúde da Universidade de Aveiro
DoS	Denial of Service
RBAC	Role Based Access Control
HTTP	Hypertext Transfer Protocol
VPN	Virtual Private Network
HTTPS	Hypertext Transfer Protocol Secure
XSS	Cross-site scripting
MFA	Multi-Factor Authentication
CSRF	Cross-Site Request Forgery
NMEC	Student ID Number
UC	Curricular Unit
FAQ	Frequently Asked Questions

CHAPTER 1

Introduction

1.1 Context and Motivation

This curricular year, the course of Medicine opened for the first time in University of Aveiro (UA). [1] Universidade of Aveiro is a pioneering institution in the field of medical education in Portugal, having been one of the first countries to implement the EPA model in medical education. The EPA model is a competency-based approach that focuses on the practical application of knowledge and skills in real-world clinical settings. This model aims to ensure that medical students are well-prepared for their future roles as healthcare professionals. The EPA model is designed to provide a structured framework for assessing and evaluating the competencies of medical students. It emphasizes the importance of entrustment decisions, where supervisors determine whether a student is ready to perform specific tasks or activities independently. This approach allows for a more personalized and tailored learning experience, ensuring that students receive the necessary support and guidance throughout their training. [2] Currently, this type of assessment is all done in a single sheet of paper, that the students bring to each evaluation hand in to their tutors, who then evaluate the autonomy of the student in each Micro Entrustable Competency (MEC). In this sheet of paper, the tutor should be able to write informative feedback to the student, so that the student can improve in the next evaluation. However, this is not always possible, due to the constraints of the paper format, which does not allow for a lot of information to be written in a single sheet. This is where the need for a digital platform arises, to allow for a more efficient and effective way of assessing and evaluating the competencies of medical students.

In response to this need, the course of Medicine proposed the development of a digital platform that replaces the outdated paper-based system and enables structured, accessible, and meaningful feedback within EPA evaluations, which not only enhances the quality of student assessments but also improves the overall learning experience by allowing tutors to provide timely and detailed feedback. This platform, named **EvalMed**, aims to support the implementation of the EPA model in a more efficient, scalable, and pedagogically effective manner.

1.2 Objectives

The main objective of this project was to design and implement a digital platform that supports the application of the EPA model in the new medical course at University of Aveiro. The platform should modernize the current assessment process by replacing the traditional paper-based system with a digital solution that enables structured, accessible, and feedback-rich evaluations while guaranteeing that this transition from paper to digital is seamless, making sure that the users think the platform is intuitive and easy to use and enjoy the experience of using it.

To achieve this, the platform must address several key system-level requirements identified during the initial design phase:

- **Multi-tenancy:** While the solution was designed to meet the specific needs of the Medicine course at University of Aveiro, it should also be adaptable for use in other courses and universities, allowing for a broader application of the EPA model in medical education.
- **Yearly Editions:** It must be possible to manage different editions of the same course across academic years (e.g., Clinical Medicine I 2025, 2026…), with independent sets of evaluations, tutors, and EPA structures for each.
- **Longitudinal Student Tracking:** The system should track each student's progress over time, across different curricular units and academic years, maintaining continuity in their EPA-based development and allowing both students and tutors to monitor growth and areas for improvement.
- **Cross-Curricular Analytics:** The platform should allow for the analysis of student performance across different curricular units, enabling tutors and administrators to identify trends, strengths, and weaknesses in student competencies.

1.3 Structure of the Document

In addition to the introduction, this document is structured into four main chapters. Chapter 2 presents the state of the art, including an overview of existing solutions that support the EPA methodology, as well as their main features and limitations. Chapter 3 focuses on the system's requirements and overall architecture, covering both functional and non-functional requirements, the technological stack used, and deployment considerations such as Docker configuration and server setup. Chapter 4 provides a detailed explanation of the implementation, with an in-depth description of both backend and frontend components, authentication mechanisms, folder structures, and the main functionalities available to different types of users. Chapter 5 describes the tests and results, particularly the usability tests conducted with users, outlining the methods, analysis and conclusions drawn from these tests. It also includes load testing results, which evaluate the system's performance under stress and its ability to handle multiple concurrent users effectively.

Finally, the document concludes with appendices that include additional supporting materials and resources relevant to the development process.

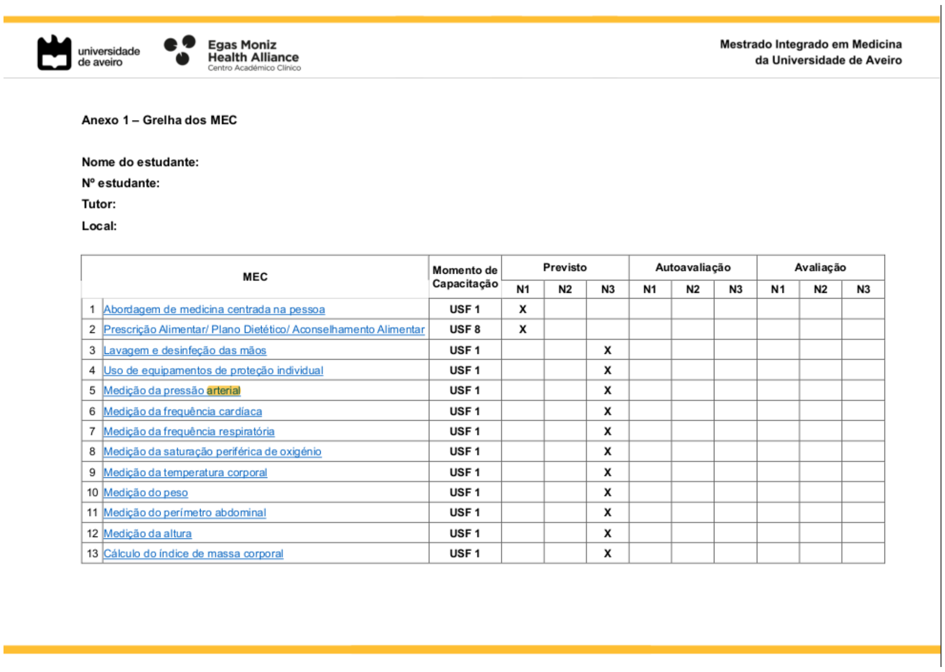
CHAPTER 2

State of the Art

2.1 Current Paper Solution for the EPA Methodology

Currently, in University of Aveiro, the EPA methodology is implemented using a paper-based system, more specifically using a CAM that is given to each student at the beginning of the year. This contains a table of all the MECs (Micro Entrustable Competencies) that the student is expected to complete during the year, along with the expected level of autonomy for each MEC. The students brings this notebook to their Family Health Unit (USF) where they perform their practical training. The students are expected to fill in their auto-evaluation for each MEC before each evaluation and then after the evalution the tutor fills in their evaluation and signs the notebook. The student then takes the notebook to the coordinator of the course who will then fill in the final evaluation for each MEC and sign it. An example of a CAM is shown in Figure 2.1. This paper-based system has several limitations, including:

- **Limited Feedback**: The paper format does not allow for detailed feedback, as there is limited space for comments and suggestions. This makes it difficult for students to understand their strengths and areas for improvement.
- **Lack of Structure**: The paper-based system does not provide a structured way to track progress or analyze performance over time. Students and tutors have to rely on manual tracking, which can lead to errors.
- **Inconvenience**: Carrying a physical notebook can be difficult, especially when students need to access it frequently during the whole year. This can lead to lost or damaged notebooks, resulting in incomplete evaluations.


universidade de aveiro

Egas Moniz Health Alliance
Centro Académico Clínico

Mestrado Integrado em Medicina
da Universidade de Aveiro

Anexo 1 – Grelha dos MEC

Nome do estudante:
Nº estudante:
Tutor:
Local:

	MEC	Momento de Capacitação	Previsto			Autoavaliação			Avaliação		
			N1	N2	N3	N1	N2	N3	N1	N2	N3
1	Abordagem de medicina centrada na pessoa	USF 1	X								
2	Prescrição Alimentar/ Plano Dietético/ Aconselhamento Alimentar	USF 8	X								
3	Lavagem e desinfeção das mãos	USF 1			X						
4	Uso de equipamentos de proteção individual	USF 1			X						
5	Medição da pressão arterial	USF 1			X						
6	Medição da frequência cardíaca	USF 1			X						
7	Medição da frequência respiratória	USF 1			X						
8	Medição da saturação periférica de oxigénio	USF 1			X						
9	Medição da temperatura corporal	USF 1			X						
10	Medição do peso	USF 1			X						
11	Medição do perímetro abdominal	USF 1			X						
12	Medição da altura	USF 1			X						
13	Cálculo do índice de massa corporal	USF 1			X						

Figure 2.1: Example of a CAM used in the Medicine Course at University of Aveiro.

2.2 Existing Software Solutions for the EPA Methodology

Before the development of our software, we conducted a thorough review of existing software solutions that are used throughout the world to support the EPA methodology or other performance assessment methods in the medical field. Here, we will go over the most relevant software solutions that we found, highlighting their features and limitations:

- **SIMPL** [3]: SIMPL is a mobile platform developed to support surgical training programs by allowing residents to log operative cases and receive structured feedback from attending surgeons. It uses a three-question operative assessment, including autonomy scoring via the Zwisch scale. It supports real-time feedback and allos the user to have access to statistics on their performance. However, it is limited to surgical training and does not support the full range of EPA-based assessments.
- **EPA Portfolio** [4]: EPA Portfolio is a subscription-based platform that digitalized the Competency-Based Medical Education by creating a web app that can also be used as a mobile app. The platform is one of the most used in the world, being used in over 60 hospitals with 15,000 users. It allows the user to track their progress, receive feedback, and manage their learning objectives.
- **EPA Tracking Tool** [5]: The EPA Tracking Tool is a web-based platform that allows users to track their progress in the EPA framework. It's unofficial meaning it's not affiliated with any official medical schools, so its usage come from students who add their own EPAs and track their progress. This can be quite tiresome as it requires manual input of data and does not provide structured feedback or assessment. It is primarily used for personal tracking rather than institutional use.

- **Moodle Platform (E-Learning)** [6]: While Moodle is not specifically designed for EPA-based assessments, we thought it was worth mentioning as it is widely used in educational institutions for most types of learning. It is an open-source learning management system that allows educators to create and manage courses, track student progress, and provide feedback. However, since in the EPA methodology the communication between the supervisor and the trainee is crucial (1:1), manipulating a moodle platform to support this methodology is not ideal. It lacks the specific features needed for structured EPA assessments, for example a student cannot easily ask for feedback from their supervisor, and the supervisor cannot easily provide feedback on specific tasks or competencies.

2.2.1 Major Features of Existing Solutions

The existing solutions we reviewed have several common features that are beneficial for EPA-based assessments:

- **Progress Tracking**: Most platforms allow users to view their overall progress across EPAs or competencies, often with dashboards or summary views. This is essential for understanding general performance over time.
- **Per-MEC Analysis**: Some platforms also allow users to analyze each MEC (Micro Entrustable Competency) or EPA individually, helping students and tutors identify specific strengths and areas for improvement at a granular level.
- **Feedback Mechanisms**: There are solutions that include features for tutors to provide feedback on student performance, which is essential for the EPA methodology.
- **Statistics and Reporting**: Some platforms offer statistics and reporting features that allow users to analyze their performance over time.
- **Mobile Accessibility**: Several solutions provide mobile applications or responsive web designs, enabling users to access their assessments and feedback on the go.
- **User Management**: Most platforms include user management features, allowing administrators to manage students, tutors, and other stakeholders effectively.

2.2.2 Limitations of Existing Solutions

After reviewing these existing solutions, we identified several key limitations that our software aims to address:

- **Reevaluation Requests**: Most existing platforms do not allow students to request reevaluations of a specific MEC. This is essential since in this methodology, students are expected to take an active role in their learning and assessment. Our software includes a feature that allows students to request reevaluations of specific MECs, enabling them to demonstrate improvement and receive additional feedback.
- **Inadequate Support for Multi-Year Logic**: None of the reviewed platforms fully handle multi-year curricular and course continuity. Our software incorporates multi-year support to accommodate longitudinal tracking and curriculum progression allowing students to see their progress over time and go back to previous years to review any MECs they have completed.

- **No Built-in USF Management**: Most solutions do not include a way to manage or locate student's health units, which are a crucial part of practical medical training. Our system integrates a USF management features, allowing to add a specific USF to each student, which can then be used to filter
- **Offline Functionality**: Many platforms depend on constant connectivity. Our solution tolerates limited offline use, enabling basic functionalities even when internet access is temporarily unavailable which can be crucial in places like a hospital where internet connectivity may be unreliable.
- **Protocol Uploads and Documentation**: Most reviewed platforms do not allow uploading structured documentation or protocols in PDF form. Our solution allows administrators to upload protocols that can then be accessed by students and tutors, ensuring they have the necessary resources for their assessments. This will make the transformation from paper-based to digital assessments less cumbersome and more efficient.

CHAPTER 3

Requirements of the System and the Architecture

3.1 Requirements of the System

When we took on this project, there were 6 main requirements proposed by the client that we had to achieve, those being:

1. The product must have a **mobile app version** available for both tutors and students
2. The backend must be **scalable** and allow to store evaluation data **in a secure way and in conformity with the GDPR**
3. The product must **facilitate tutor feedback with an on-demand structure**, allowing students and tutors to see progress and allow them to identify areas of improvement
4. The product must include **a performance analysis module**, showing graphs and progress indicators
5. The product must be **functional while offline**, allowing actions to be performed while offline to be synchronized afterwards
6. The product must have a **permission / access system**, ensuring that different users have different persmissions and only some users can see and insert evaluations.

In order to develop our product, we need to break down these main requirements into smaller, more objective requirements which we could complement with acceptance criteria, following the usual lifecycle of a project of this nature.

In order to get to these requirements, we must first trace back to **who uses our product** and **for what purpose**, and for that we will use Personas, Scenarios and User Stories.

Building Personas helps us in worldbuilding, which is especially important in this case, since we have little to no exposure with Medicine, or any Medicine-related course.

From there, we create scenarios for these users, and we create user stories based on these scenarios, and finally we extract our requirements from those user stories.

3.1.1 Personas, Scenarios & User Stories

The main requirements, as well as several meetings with our clients, allowed us to extrapolate 3 actors that would our product:

1. **Tutor**: Responsible for evaluating students and providing feedback
2. **Student**: Responsible for completing the practical evaluations and receiving feedback
3. **Administrator**: Responsible for managing the tutors and students, and ensuring that the platform is updated and working correctly

For each of these actors, we developed a persona that would help us elaborate user stories with more ease.

Photo	
Fictional Name	Luísa Laranjeiro
Environment	Luísa completed her MSc in neurology and was invited to tutor the University of Aveiro students in her USF due to the creation of the Medicine degree. She is used to technology, altough can struggle a bit with some interfaces. She tutors a few students and try their best to convey their knowledge to them and guarantee they are successful.
Job Title / Major Responsibilities	Teach and evaluate students during their trips to the USF.
Demographic	39 years old, lives in Aveiro, Clinical Medicine Practician
Goals and Tasks	• Ensure the practical evaluations of their students are scheduled. • Complete the practical evaluation of their students by filling an evaluation form. • Guarantee students have feedback on their evaluations so they can reflect on past work.

Tutor Persona

Photo	
Fictional Name	Maria do Mar
Environment	Maria is a student that loves health-related science and dreams of being a surgeon. She's resilient and strives to achieve her goals. To study, she mainly uses digital platforms to organize her work due to the high volume of theoretical coursework.
Job Title / Major Responsibilities	Complete practical evaluations with a good score.
Demographic	19 years old, Medicine student at the University of Aveiro
Goals and Tasks	• Study all MECs procedures and get good grades in the practical evaluations. • Make sure not to miss any evaluation days. • Keep track of her grades and past work so she can improve. • Adapt to the feedback being given so she can improve.

Student Persona

Photo	
Fictional Name	Mário Silva
Environment	Mário is the head teacher of Clinical Medicine I at the University of Aveiro. He's somewhat used to working with technology. He manages the course and assigns tutors to students. He wants each tutor to be as active as possible in a student's evaluation.
Job Title / Major Responsibilities	Guarantee the tutors and students don't face any complications while using the application and that all students are receiving enough feedback to reach their potential.
Demographic	52 years old, Medical Teacher
Goals and Tasks	• Handle the management of tutors and students. • Ensure the application's information is always updated. • Check an overview of the feedback being given and if tutors are communicating with their students.

Administrator Persona

With these personas in mind, we created the following scenarios:

Medical Student Scenario (Maria – First-Year Medicine Student)

Maria is a first-year Medicine student at Universidade de Aveiro, enrolled in a subject taught using the **EPA method**. She uses **EvalMed** to track her progress and monitor her performance in this subject.

Previously, evaluations were done on a **single piece of paper**, making it difficult to review past results and organize feedback. Now, with **EvalMed**, she can **easily access her complete evaluation history**, analyze her **strengths and weaknesses**, and understand which **MECs (Medical Entrustable Competencies)** she needs to improve.

Additionally, she has difficulty completing the **"Otoscopy Exam"** with an adequate level of autonomy, so she accesses the **educational content** present in the details of the specific MEC, helping her **learn and focus on areas where she needs the most support**.

Tutor Scenario (Luísa – Medical Tutor)

Luísa is a tutor at Universidade de Aveiro, responsible for evaluating and mentoring students using the **EPA method**. She uses **EvalMed** to efficiently track student progress and structure upcoming **USF** based on student performance.

Before EvalMed, students had to **physically bring paper evaluations**, which limited preparation. Now, she can **instantly access her students evaluations and self-assessments**, allowing her to identify their **biggest weaknesses and strengths**. This helps her **prepare more effective USF sessions** and provide **targeted feedback** to help students improve.

Administrator Scenario (Mário – Clinical Medicine I Supervisor)

Mário is a professor at **Universidade de Aveiro**, overseeing the **Clinical Medicine I** course. He uses **EvalMed** to **monitor student and tutor performance** and ensure that the course runs smoothly. He wants to update **MEC procedures** with the **latest medical guidelines** so that students and tutors always work with the most **current and accurate information.**

Using **EvalMed's version control system**, he can **edit existing MECs, add new ones, and review past versions if needed**. This ensures **transparency** and maintains a record of all updates for reference. After that he wants to check the **feedback given by tutors** to students to ensure that they are **receiving enough feedback** to reach their potential.

He goes to the dashboard and checks the **feedback overview** to see if the **communication between tutors and students** is effective and the number of feedback being given by each tutor.

And finally, from those Scenarios we extracted some User Stories. A User Story contains a context of use for a certain actor within a certain context / scenario, and each User Story has a **acceptance criteria** that tells us if that user story has been fulfilled or not.

3.1.1.1 Student User Stories

As a student,
I want to check the observations made by the tutor during the practical evaluation,
So that I can improve my technical and theoretical knowledge.

Acceptance Criteria:

- **Given** a student wants to check observations made by the tutor during the practical evaluation
- **When** the student checks the details of the evaluation performed in the app
- **Then** the system will provide all details about that evaluation, including observations done by the tutor

---

As a student,
I want to receive notifications about new grades and feedback posted by my tutor,
So that I can always be updated regarding my progress.

Acceptance Criteria:

- **Given** a student wants to receive notifications about new grades or feedback
- **When** a tutor publishes feedback or an evaluation into the system
- **Then** the student will receive a push notification stating that this evaluation/feedback has been posted

---

As a student,
I want to check the history and statistics of my evaluations in each of the MECs,
So that I can see my evolution across the year.

Acceptance Criteria:

- **Given** a student wants to check the history and statistics of their evaluations in each of the MECs
- **When** the student selects a MEC in the search page
- **Then** the system will provide a history of evaluations, with the date and evaluation of that MEC.

As a student,
I want to check my evaluation calendar,
So that I can adequately prepare for each evaluation moment.

Acceptance Criteria:

- **Given** a student wants to check their evaluation calendar
- **When** the student selects the calendar page in the navbar
- **Then** the system will provide a calendar providing next evaluations and trips to a USF labeled, as well as the next relevant event highlighted.

As a student,
I want to access educational material (like videos or documentation) regarding each MEC,
So that I can deepen my knowledge before my practical evaluation.

Acceptance Criteria:

- **Given** a student wants to access educational material regarding a certain MEC
- **When** the student clicks the intended MEC to watch educational content
- **Then** the system will provide educational content related to that MEC at the bottom of the page

---

As a student,
I want to request that some MECs are reevaluated in the next USF session,
So that I can improve my evaluated autonomy level for those MECs.

Acceptance Criteria:

- **Given** a student wants to reevaluate some MECs in the next USF session
- **When** the student submits the request using the system's interface, selecting the MECs in which they did not fulfill the expectations and that they wish to improve
- **Then** the system will send a notification to the tutor so that the tutor can accept or reject the request

---

As a student,
I want to know where my assigned USF is,
So that I can get there without any complications for my evaluations.

Acceptance Criteria:

- **Given** a student wants to know where their assigned USF is
- **When** the student presses the "locate USF" button

- **Then** the system will send a notification to the head teacher

---

As a student,
I want to self-evaluate myself on the upcoming USF evaluation,
So that I can compare my perception of my abilities to how I actually performed.

Acceptance Criteria:

- **Given** a student wants to perform a self-evaluation on the MECs that compose the next USF evaluation
- **When** the student submits the self-evaluation by filling all the fields (autonomy levels, strong points, and weak points)
- **Then** the system will store this and send a notification to the tutor saying that the student has submitted a self-evaluation

---

As a student, **I want to** see which competencies are linked to each MEC and track the competencies I gain as I complete them, **So that** I can understand my skill development and stay aware of my academic progress.

Acceptance Criteria:

- **Given** that the student has completed one or more MECs
- **When** they select the Statistics tab
- **Then** a Radar chart will show up with a percentage of each competency obtained.

As a student, **I want to** download PDF files of a specific MEC with the respective procedures and other details, **So that** I can study on the go and don't have to log into the application.

Acceptance Criteria:

- **Given** the student is going to be without their phone for a field trip
- **When** they go to a certain MEC details page
- **And** click on the button "Download PDF"
- **Then** the system will download a PDF file into the student's device with all the MEC details.

3.1.1.2 Tutor User Stories

As a tutor,
I want to evaluate my student as they perform their practical evaluation,
So that they can have an up-to-date evaluation on their CAM.

Acceptance Criteria:

- **Given** a tutor wants to evaluate their student as they perform their practical evaluation
- **When** the tutor fills the MEC evaluation, the strong points, and weak points
- **Then** the system will store this and send a notification to the student saying that the tutor has submitted an evaluation

---

As a tutor,
I want to check the evaluation history regarding a specific student,
So that I can have an overview of their evolution.

Acceptance Criteria:

- **Given** a tutor wants to check the evaluation history regarding a specific student
- **When** the tutor selects the student and checks the evaluations tab
- **Then** the system will show all past and future evaluations for that student

---

As a tutor,
I want to change the student of whom I'm seeing information about to another who I am tutoring,
So that I can have easy access to information about all my students.

Acceptance Criteria:

- **Given** a tutor wants to change the student of whom they're seeing information about to another
- **When** the tutor selects the new student they wish to see information about
- **Then** the system will update all visible information to information relative to the new student

---

As a tutor,
I want to know when all the upcoming evaluations take place,
So that I can balance my personal and professional life and avoid delays and absences.

Acceptance Criteria:

- **Given** a tutor wants to know when all the upcoming evaluations are
- **When** the tutor checks the calendar or the evaluations tab

- **Then** the system will display the upcoming evaluations as well as the past ones, highlighted in the calendar and labeled, and listed in the evaluation page

---

As a tutor,
I want to correct a previous evaluation I submitted,
So that I can correct a mistake I made in the submission.

Acceptance Criteria:

- **Given** a tutor wants to correct a previous evaluation they submitted
- **When** the tutor submits the corrected version of the evaluation
- **Then** the system will notify the student that the evaluation has been revised

---

As a tutor,
I want to analyse a reevaluation a student has requested for the next USF,
So that I can provide feedback and incorporate it in the next evaluation.

Acceptance Criteria:

- **Given** a tutor wants to analyse a reevaluation a student has requested for the next USF
- **When** the tutor opens the notification and sees the requests and selects if they want to accept or reject it
- **Then** the system will perform the changes in the database for the upcoming evaluations and notify the student about the tutor's decision

---

As a tutor, **I want to** add new MECs to specific USF evaluations, **So that** the evaluation reflects the student's advanced progress and current level of development.

Acceptance Criteria:

- **Given** that a tutor wants to add new MECs to a specific USF evaluation,
- **When** the tutor selects a USF evaluation and clicks on "Add a new MEC",
- **And** selects the MECs to include in the evaluation,
- **Then** the system must update the database to include the new MECs for upcoming evaluations and send a notification to the student informing them of the changes made by the tutor.

3.1.1.3 Administrator User Stories

As an administrator,
I want to change the tutor associated with a student,
So that if the tutor cannot attend, the evaluation can still occur with a different tutor.

Acceptance Criteria:

- **Given** the administrator wants to change the tutor associated with a student
- **When** the administrator selects the intended student and the tutor they wish to be associated with that student
- **Then** the system will send a notification to both the tutor and the student saying the tutor has been changed and perform the changes in the database

---

As an administrator,
I want to add/edit the MEC,
So that the MEC are always updated and aligned with the current course guidelines.

Acceptance Criteria:

- **Given** the administrator wants to add or edit a MEC description
- **When** the administrator completes the edition or addition of the MEC by filling all required fields
- **Then** the system will add the MEC to the database and it can be evaluated by the relevant years

---

As an administrator,
I want to be able to set the expectations for each MEC, for each year, as well as the time in which those expectations should be met (ex: N1 for MEC 1.1.1 at USF1),
So that I can help the tutors evaluate the students across the year.

Acceptance Criteria:

- **Given** the administrator wants to set the expectations for each MEC, for each year as well as the time in which those expectations should be met
- **When** the administrator selects the MEC and edits its details
- **Then** the system will update these data and apply them to everyone

---

As an administrator,
I want to be able to close the year and begin a new year, saving/resetting all the collected information regarding grades,
So that I can keep everything organized between years and be able to follow a yearly routine.

Acceptance Criteria:

- **Given** the administrator wants to close the year and begin a new year
- **When** the administrator selects the “End Year” option
- **Then** no new evaluations will be accepted, no MEC requests will be possible, it will be possible for tutors to evaluate the assiduity and interest of each student, and students will see a year overview on the main page

---

As an administrator, **I want to** be able to upload an Excel file for a specific academic year containing student, tutor, and USF information, **So that** I can load this data into the system and either automatically perform all necessary student–tutor–USF assignments or handle them manually.

Acceptance Criteria:

- **Given** the administrator wants to create a new academic year,
- **When** the administrator selects the “Students and Tutors” tab,
- **And** clicks on “Upload Excel File”,
- **And** maps the correct columns from the file to the required system labels
- **Then** the system must load the students, tutors, and USF evaluations into the system and make the data available for assignment and management

---

As an administrator, **I want to** be able to download an Excel file for a specific academic year containing student information (e.g., name, Student ID Number (NMEC), current grade, number of completed MECs), **So that** I can easily analyze and archive academic data for administrative purposes.

Acceptance Criteria:

- **Given** the administrator is in the Home page and wants to download information about the selected course
- **When** the administrator clicks on the “Download Excel File”
- **And** selects the specific information they want in the Excel file
- **And** chooses the order they want the information to be shown
- **Then** the system will download an Excel file into the administrator’s device with all the requested information about the students

---

As an administrator, **I want to** be able to add/edit evaluation days/hours, **So that** the platform reflects the correct and up-to-date schedule for evaluations.

Acceptance Criteria:

- **Given** the administrator wants to edit an already planned evaluation
- **When** they click on the “Program Curricular Unit (UC)” tab

- **And** click on the editable fields to change date and start/end time of a specific evaluation
- **And** click on “Submit” button
- **And** confirm that the changes made are correct
- **Then** the system will update the database with the new schedule and send notifications to all students

---

As an administrator, **I want to** dismiss a doctor from being a tutor, **So that** the system accurately reflects current teaching assignments and responsibilities.

Acceptance Criteria:

- **Given** that the administrator wants to dismiss a tutor from a specific student
- **When** they click on the “Students and Tutors” tab
- **And** search for the student’s name
- **And** click on the “Dismiss Tutor” button
- **And** confirm the dismissal
- **Then** the system will remove the tutor from that student and the option to assign a new tutor will appear

---

As an administrator, **I want to** assign a new tutor to a specific student, **So that** the student is always linked with an active and appropriate tutor for their academic guidance.

Acceptance Criteria:

- **Given** that the administrator wants to assign a new tutor to a student
- **When** they click on the “Students and Tutors” tab
- **And** search for the student’s name
- **And** click on the “Assign Tutor” button
- **And** select a doctor from the list of available tutors
- **And** confirm the assignment
- **Then** the system will link the selected doctor as the student’s tutor and display the new tutor’s info in the student’s profile

---

As an administrator, **I want to** edit specific information of a student or tutor, **So that** the data remains accurate.

Acceptance Criteria:

- **Given** that the administrator wants to update the information of a student or tutor
- **When** they access the “Students and Tutors” tab
- **And** search for the name of the desired student or tutor
- **And** click the “Edit” button next to their name

- **And** modify specific fields (e.g., name)
- **And** confirm the changes
- **Then** the system updates the information immediately and displays a success message

---

As an existing administrator, **I want to** be able to add or remove other administrators, **So that** I can manage who has administrative access to the platform.

Acceptance Criteria (Adding):

- **Given** that an administrator wants to grant admin access to another user
- **When** they navigate to the "Administrators" tab
- **And** click the "Add Administrator" button
- **And** insert the email and name of the new administrator
- **And** confirm the information
- **Then** a confirmation message is displayed

Acceptance Criteria (Removing):

- **Given** that an administrator wants to revoke admin access from another user
- **When** they navigate to the "Administrators" tab
- **And** locate the user in the list of current administrators
- **And** click the "Remove" button
- **And** confirm the action in a confirmation prompt
- **Then** the system removes the administrator role and shows a confirmation

---

As an administrator, **I want to** view general statistics about the course, **So that** I can monitor overall academic performance, student progress, and identify difficult MECs.

Acceptance Criteria:

- **Given** that the administrator wants an overview of the course performance
- **When** they navigate to the "Statistics" section
- **Then** the system will display:
  - The current number of students enrolled
  - The average grade across the course
  - The percentage of students that have completed each MEC
  - The average number of MEC evaluations in each USF
  - The average number of feedback/reevaluation requests per month

---

With these user stories laid out, we were able to build a User Case diagram:

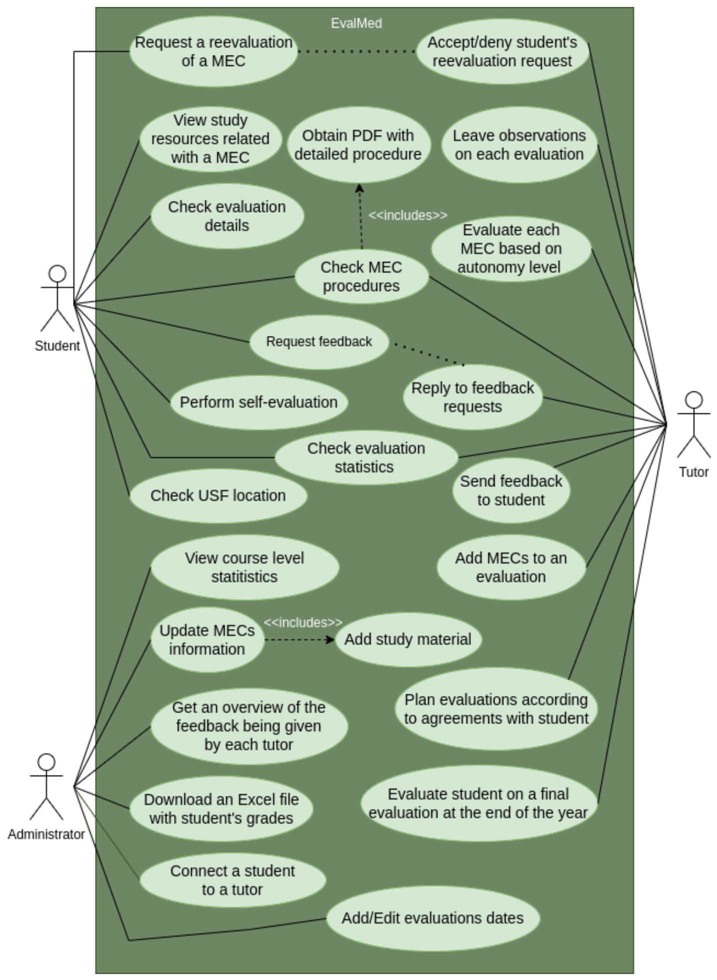

EvalMed
Request a reevaluation of a MEC
Accept/deny student's reevaluation request
View study resources related with a MEC
Obtain PDF with detailed procedure
Leave observations on each evaluation
Check evaluation details
<<includes>>
Evaluate each MEC based on autonomy level
Check MEC procedures
Student
Request feedback
Reply to feedback requests
Perform self-evaluation
Check evaluation statistics
Tutor
Send feedback to student
Check USF location
View course level statitistics
Add MECs to an evaluation
Update MECs information
<<includes>>
Add study material
Plan evaluations according to agreements with student
Get an overview of the feedback being given by each tutor
Administrator
Download an Excel file with student's grades
Evaluate student on a final evaluation at the end of the year
Connect a student to a tutor
Add/Edit evaluations dates

Figure 3.1: Use Case Diagram

3.1.2 Requirement Elicitation

With our 6 main directives and the work performed to generate user stories, as well as meetings with our clients, we were able to create a set of functional and non-functional requirements, which will follow:

3.1.2.1 Functional Requirements

1. **User Management & Authentication**
   - The system must have authentication via UA's Identity Provider (IdP) using WSO2
   - The system must support different permission levels (student, tutor, administrator).
   - The administrator should be able to associate a student with a tutor.
2. **Evaluation & Feedback Management**
   - The system must allow the tutors to submit the USF evaluation for a student.
   - A tutor must be able to add observations to an evaluation (strong and negative points).

- A student must be able to check all completed evaluation grades and observations.
- The student must be able to request feedback from the tutor for either the operation of a MEC or feedback on a previous/upcoming evaluation.
- The tutor must be able to reply to a feedback request by a student.
- The system should allow students to ask for reevaluations of a specific MEC on an upcoming evaluation.
- The system should allow tutors to accept or decline a student's request for reevaluation.
- The system must only allow the USF4 and USF8 to be submitted if given feedback.
- The student needs to be able to insert their self-evaluation regarding their autonomy on performing each MEC a few days before the next evaluation.
- The student needs to be able to insert their self-evaluation regarding strong points and weak points in each of the MECs.

3. **Tutor Evaluation Management**
   - The system must allow tutors to revise and correct previously submitted evaluations.
   - The system must allow tutors to add new MECs to a scheduled USF evaluation for a student.
4. **Student Progress & Performance Tracking**
   - The system must present graphs and progress bars relative to a student's progress.
   - The system should automatically calculate the CAM score for each student according to the established criteria.
   - The system must be able to identify the strong points and weak points of each student and the areas of improvement.
   - The system should show which MECs can be improved and which ones have the maximum grade already.
   - The system must allow the tutor to evaluate, at the end of the year, the student based on punctuality, attendance, and interest.
5. **Competency Tracking**
   - The system must display the competencies associated with each MEC.
   - The system must visualize student competency acquisition (e.g., using a radar chart).
6. **Location & USF Mapping**
   - The system must allow students to view the location of their assigned USF on an interactive map.

- The system must allow users to access an external link to Google Maps for directions to the USF.

7. **Version History & Data Management**
   - The system should allow the administrator to update MECs description/procedure.
   - The system should allow the administrator to add study material to each MEC.
8. **Search & Filtering Capabilities**
   - The system must allow a search for MECs, filterable by:
     - Expected autonomy levels (N1/N2/N3)
     - Acquired autonomy levels
     - Medical competencies related to each EPA or MEC
     - Time of evaluation
     - Place of evaluation
9. **Calendar & Scheduling**
   - The system must be able to display a calendar tailored to the currently signed-in user with the next evaluations.
10. **Administrative Year Management**
    - The system must allow administrators to edit or reschedule evaluation dates and times.
    - The system must allow administrators to upload Excel files to import student, tutor, and USF data.
    - The system must allow administrators to end the academic year, freezing all current data.
    - The system must allow administrators to start a new academic year with updated configurations.
11. **Reporting & Documenting**
    - The system must be able to present each MEC's procedure in a PDF file and allow its download.
    - The system shall allow administrators to download an Excel file of all of the students' grades.
12. **Notifications & Alerts**
    - The system must be able to send push notifications to users for:
      - Evaluation updates
      - MEC updates
      - Upcoming evaluations
      - Start/end of year notifications

  - Feedback notifications

13. **Role & Access Management**
    - The system must allow administrators to add or remove other administrator accounts.
    - The system must allow administrators to edit student and tutor information (e.g., name, email, assigning tutor).

3.1.2.2 Non-Functional Requirements

1. **Performance**
    - The system must maintain consistent response times as user traffic increases.
    - The system must efficiently handle a high number of requests, ensuring low latency and high performance even under increasing load.
2. **Security & Privacy**
    - The system must comply with **GDPR** guidelines, ensuring the protection of student data.
    - User's inputs should be sanitized to avoid common security problems such as Cross-site scripting (XSS), Structured Query Language (SQL) injection or Cross-Site Request Forgery (CSRF)
    - Role-based access control (RBAC) must be enforced to grant different permission levels (students, tutors, administrators).
    - Authentication must be handled via UA's IdP using secure authentication methods.
3. **Usability & User Experience**
    - The system must adapt to smaller screens, such as mobile phones and tablets.
    - The user interface must follow a **consistent and simple design** to avoid confusion.
    - Forms must provide **real-time visual feedback** when users make errors.
    - Icons must be **representative of their function** and consistent across all pages.
    - The system should have a support section with **Frequently Asked Questions (FAQ)s and a contact form** for users to report issues.
    - The system should have a guide for first-time users to help them **navigate the platform** in case they have any doubts.
4. **Maintainability**
    - The system code must be **simple and easy to understand** (and commented when necessary) to allow future developers to modify the system when required.

- All API endpoints must be **documented**, providing details about their purpose and usage within the application.
- A **logging system** must be implemented to record all significant errors and warnings occurring in the system.
- Logs must include a **timestamp and the source of the error** (affected service).
- The **logging system must be centralized**, consolidating logs from all system components.

5. **Scalability**
   - The system must accommodate an **increasing volume of data** without significant performance degradation.
   - The system must be **easily scalable** to support a growing number of users and data.
   - Database queries should be optimized for fast access and indexing should be used to handle large datasets.
6. **Reliability & Fault Tolerance**
   - Some system functionalities must be operational even in the absence of an internet connecting.
   - Specific changes made offline should be synchronized with the server once the connection is reestablished.

3.2 Architecture

With our requirement elicitation we had an idea of what kind of system we would want - we wanted a product that had a front-end that could communicate with the user, as well as communication with a database that would store all needed information.

As such, we had 2 domains that we needed to make decisions on.

3.2.1 Backend

First, we needed to decide on what kind of architecture we wanted. We wanted something that could be decoupled from the frontend, so that we could perform changes indepedently, but also something that would not be too complex as to waste precious time in setting things up. In the end, a **Service-Oriented Architecture** proved to be the best fit for our needs.

We needed our services to specialize in different aspects of our product. We identified a few key aspects of our product:

- User Management
- Authentication
- EPA Management
- Notifications Management
- Feedback Management
- Course Management

As such, those became our needed services. Because the students, tutors and administrators have different characteristics and roles, we separated the user management into 3 different services.

For the Authentication, a requirement imposed on us was using the UA's IdP as the authentication method. We faced many issues with this, and our solution was more of a workaround, which we will describe in the implemetation chapter in section 4.3

Similarly, for notifications, we decided that the most effective way to implement notifications would be by sending an SMS / Whatsapp message to the users whenever they got a notifications. However, out of the many alternatives we explored, none met our price requirement, therefore we had to pursue different avenues. We ended up settling for Push Notifications, as well as Email Notifications. 2 new dependencies would arise.

For the push notifications, we opted to use Firebase Cloud Messaging (FCM). It's a Google solution, highly dependable, that is used by many companies. The notification system is also free which was a big plus.

For the email notifications, used primarily for login purposes, we used JavaMailSender in order to send an email to the recepient, using an email that we created for the purpose of our solution (evalmed2025@gmail.com).

Because our system contained many features and needs, we expanded beyond the core services we had, and as such, we have the following services:

- AdministratorService - deals with all Administrator-related operations, that only an administrator could perform.

- CourseService - deals with all the details regarding courses, their details, evaluations, etc.
- EmailService - responsible for sending emails to people about feedback, reevaluations, or others
- EPAService - deals with all EPA-detailed information, such as managing and finding all EPA_Nested and MECs within an EPA.
- FCMService - responsible for sending push notifications to the different devices that allowed them
- FeedbackService - deals with all feedback business logic
- PersonService - deals with the current person / identity of a user, allowing to update the Person's details or set a new profile picture
- ReevaluationService - deals with all reevaluation business logic
- StudentService - deals with student-specific operations, such as querying for students by Course, id, or creating an evaluation for a group of students
- TutorService - deals with tutor-specific operations, such as managing feedback and evaluations, or evaluation details.
- UserService - deals with all user-specific information, like notification tokens and details about the current user
- WelcomeEmailService - responsible for sending a welcoming email when an account is created on the system.

3.2.2 Databases

Another decision that we would need to make would be how to store our data. Since we detected clear relationships between users, evaluations, universities and EPA's, it became clear to us that we needed to use a Relational Database to represent all these relationships. However, for the MEC's, we identified that their structure, while similar, was different in many cases, presenting different number of steps in procedures, therefore we opted to store the MEC's in a separate document-based DB.

3.2.2.1 MySQL

For the majority of our structured data, we opted to use MySQL, a widely used relational database management system. MySQL was an easy choice because it allowed us to define clear relationships between entities such as users, evaluations, universities, and EPA's. Its support for SQL provided a straightforward way to query and manipulate data, which was essential for many of our use cases. Additionally, MySQL has great reliability and scalability, ensuring that it could handle our system's requirements both during development and in production, while also giving us guarantees for the future.

3.2.2.2 MongoDB

For the MEC's, we needed a database capable of handling flexible and dynamic structures since the number of steps and their details varied significantly between different procedures.

MongoDB, a NoSQL document-based database, was a good fit for this requirement. Its schemaless nature allowed us to store the MEC's as documents, with each document representing a unique procedure. This approach provided the flexibility we needed while ensuring that we could easily query and update the data as required.

3.2.3 Observability

In order to be able to monitor the system from the exterior, we implemented an Observability mechanism using Grafana, which captured data from Promtail + Loki and Prometheus. This dashboard includes logs and metrics from the system, in order to be able to figure out how things went wrong, and if the system is struggling.

3.2.4 Frontend

For the frontend, we also opted for a framework all of us were confortable with - React.

As we needed our product to also work while the user was offline, we needed to implement a **Service Worker**, which essentially served as a cache that could not only store whole pages, it can also store requests to the API that can be sent when the user is online for synchronization.

We also used several depedencies that helped us achieve our goals - like React-PDF and Chart.js.

In order to achieve our requirement of having a **Mobile Application**, we opted to make our website a **Progressive Web Application (PWA)**, for which we needed the Service Worker which we already had, as well as a manifest and a logo to make up the app's icon.

3.2.5 Middleware

In order to deploy our solution, we were given a machine at Institute of Electronics and Informatics Engineering of Aveiro (IEETA), however only one port was made available to us, and we needed both the frontend and backend to be reachable from the exterior. As such, we needed a middleware that would sit in the exposed port, and redirect requests to the intended components.

Our middleware is composed of a reverse proxy, Nginx, which serves as a key component in connecting the frontend and backend. Nginx acts as an intermediary, routing incoming client requests to the appropriate backend services or serving static files directly to the client. This ensures efficient communication between the frontend and the backend services, while also enabling load balancing and caching to optimize performance.

3.2.6 Containerization

To simplify both development and deployment of our application, we decided to use Docker for containerization. Docker allows us to package our application and its dependencies into lightweight, portable containers, ensuring consistency across different environments, from local development to production.

Furthermore, one of the most important parts of this approach, is that Docker streamlined our development environment by ensuring that all team members worked in identical environments. This reduced compatibility issues and prevented hundreds of hours troubleshooting environment specific problems.

3.2.7 Domain model

In Figure 3.2 we can see our ERD. This diagram represents the relationships between the different entities in our system, including users, evaluations, MECs, and USFs. Each entity is represented with its attributes, and the relationships between them are depicted with lines connecting the entities. The cardinality of each relationship is also indicated, showing how many instances of one entity can be associated with instances of another entity. Although it suffered several changes and is constantly updated as our requirements change, this diagram is a great representation of the structure of our relational database and how the different components of our system interact with each other.

3.2.7.1 MySQL

In our relational database, we have the following models:

Users - The *Person* entity serves as the base model for different types of users in the system, including *Student*, *Tutor*, and *Administrator*. Each user has attributes such as *name*, *email*, and *enabled*, among others. Students are also distinguished by their *NMEC* and are associated with both a *Tutor* and a *Course*. Tutors and Administrators inherit the common attributes from the *Person* model. Each year, a student is associated to a tutor within a course. This relationship is represented by the *StudentCourseTutor* model.

Authentication - For the One-Time Token (OTT) Login we store the logins temporarily using the *LoginToken* model, associating an email with its token

Courses and Universities - The *Course* model represents the course a *User* is enrolled. Each course is associated with a *University* and includes attributes like *name*, *startDate*, and *endDate*. The *University* entity stores information about the institution, like its *name*.

USFs and Locations - The *USF_Moment* entity captures unique moments that happen in a specific *USF*. It includes attributes like *date*, *start_time*, and *end_time*. These moments are linked to a *Course*. Additionally, the *USF_Location* model represents physical locations where these activities take place.

MECs: - The *MEC* model represents structured educational components within a course. Each MEC is linked to an *EPA_Nested* structure, which organizes hierarchical levels of evaluations. MECs also include attributes such as *name*, *version*, *hierarchy_code*, and *autonomy_level*. Each MEC is connected to one or more *Competencies* which represent soft-skills required to carry out different tasks. Every MEC is inserted in an *EPA_Nested* and its correspondent *EPA*

Evaluations and Feedback - The *Evaluation_USF* model captures evaluations of students in specific *USF_Moments*, linking students and moments. The *Evaluation_MEC* model evaluates a student's performance in a specific MEC, including their *autonomy* and *selfAutonomy*. The *Feedback* model stores feedback requests from students or tutors, along with details like *type*, *state*, *question*, and *answer*. The *AutoEvaluation* model stores the self-evaluation from a student in a given evaluation (to be submited before the evaluation takes place). The *Evaluated_In* model stores the location in which the evaluation took place. The *FinalEvaluation* model represents the evaluation performed at the end of the year, containing the levels of *attendance*, *participation* and *interest*

Reevaluations and Tutor Evaluations - The *Feedback* allows for a tutor to provide feedback to a student regarding an evaluation The *Reevaluation* entity allows students to request a reevaluation of a specific MEC, storing details like *state*, *request_time*, and *reply_time*. The *TutorEvaluation* model, on the other hand, enables tutors to provide evaluations of students, including *strongPoints* and *weakPoints*.

Notifications - The *Notification* model represents system notifications sent to users. Notifications include attributes like *title*, *description*, *type*, and *date*.

3.2.7.2 MongoDB

In addition to the relational database, we also use a MongoDB database to store *MEC* details which need more flexibility.

MEC Details - To do this we use the *MecDetails* model, it stores detailed information about MECs, such as their *name*, *version*, *autonomyLevel*, and *academicYear*. This model also includes nested structures, like a list of *ProcedureStep* (each with a *step* and *description*), *MECCompetencies* and a list of *bibliography* items. This flexible structure is better suited for the hierarchical and document-like data associated with MECs. A MEC may have resources associated with it. Those are stored within the *MEC_Resources* model. The PDF version of the MEC is stored in the *MEC_File* model

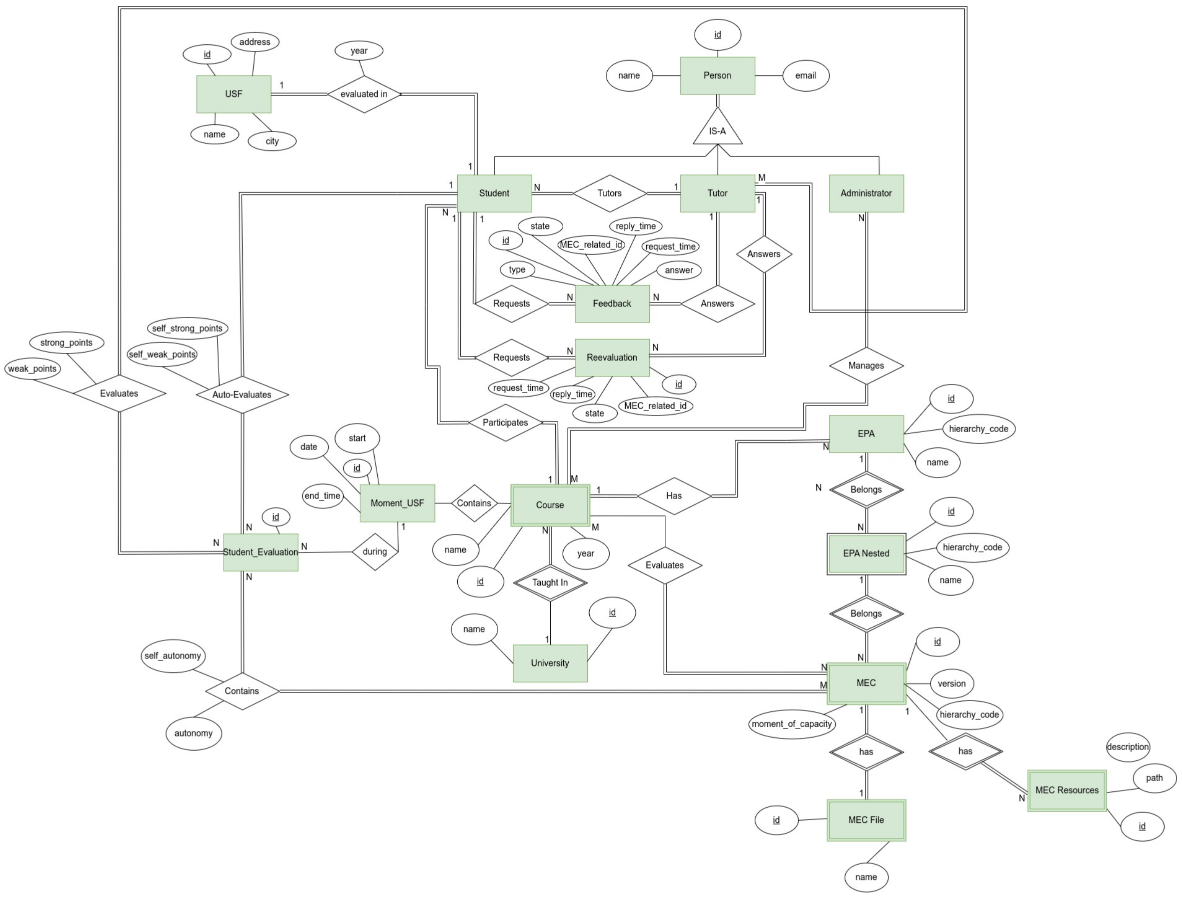


Figure 3.2: ERD (Entity Relationship Diagram)

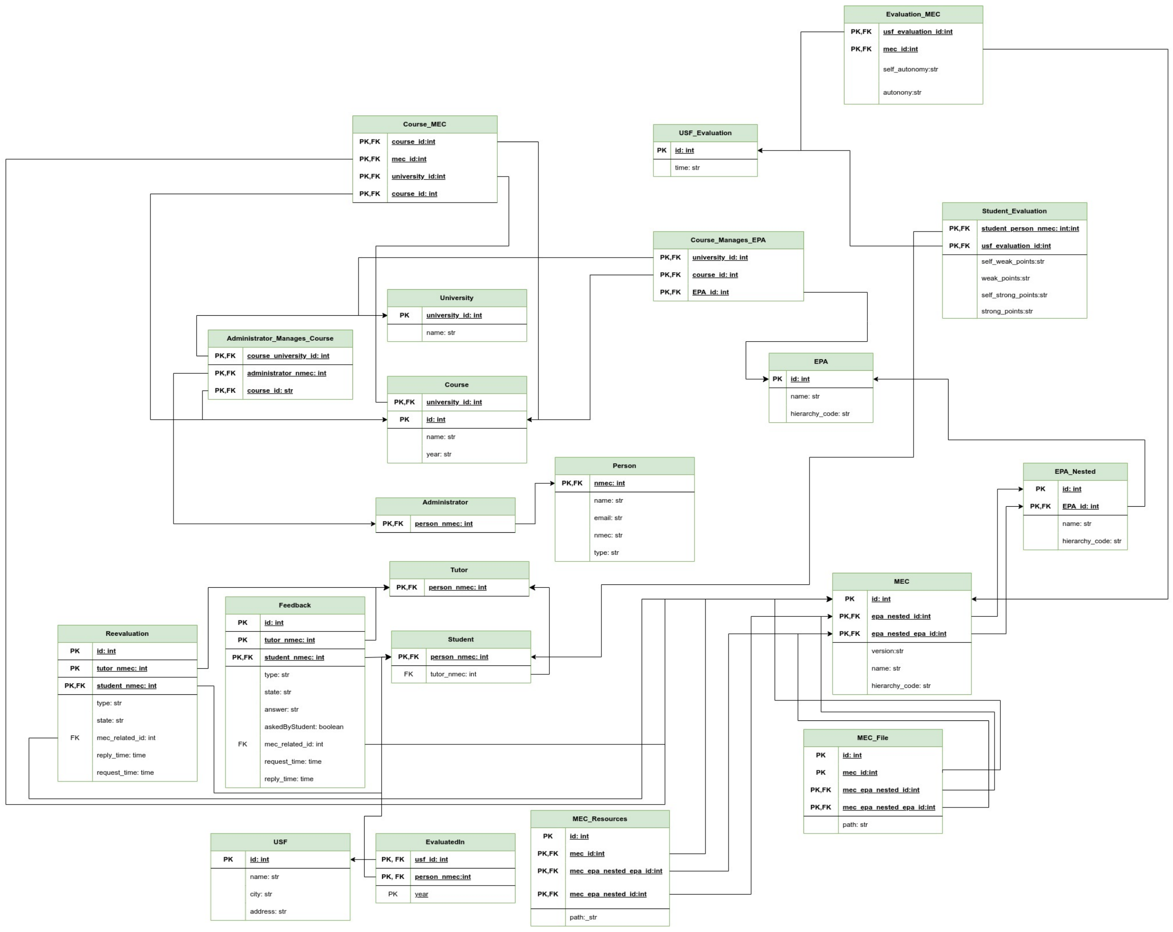


Figure 3.3: ER (Entity-Relationship)

3.2.8 Architecture Diagram

In Figure 3.4 the architecture diagram can be seen, depicting all the aspects mentioned previously, allowing our system to fulfill all requirements and be monitored from the outside.

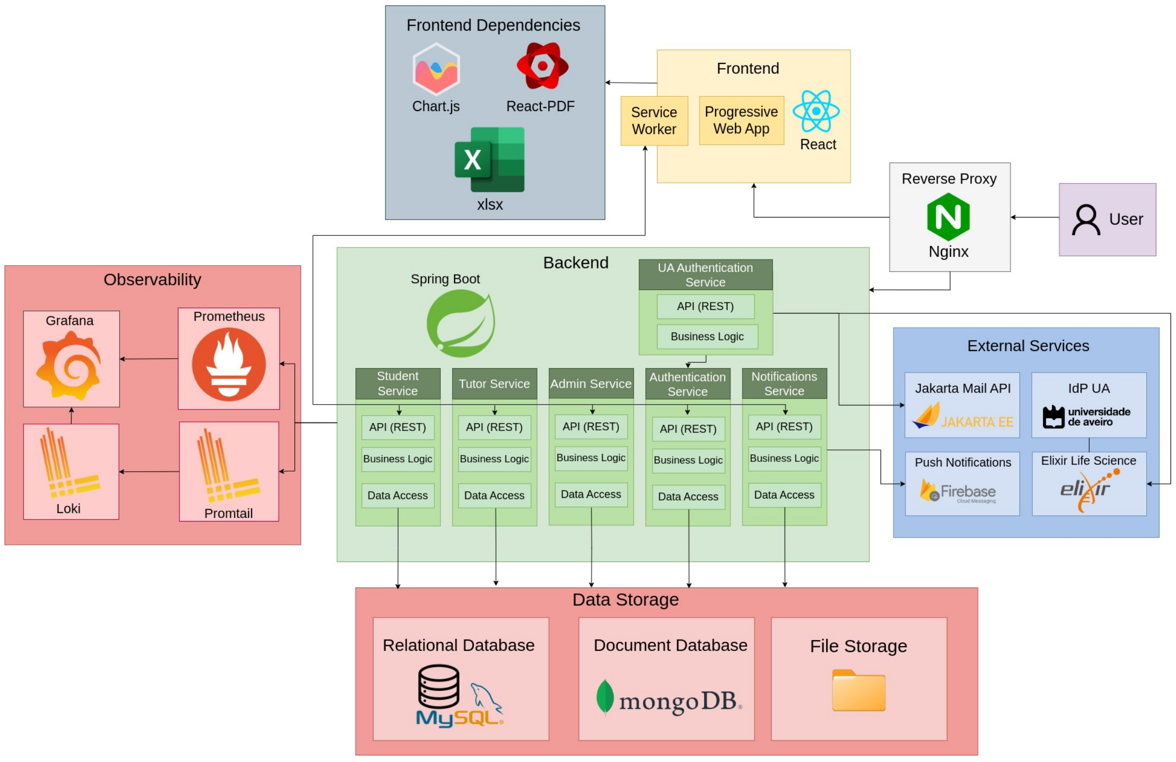


Figure 3.4: Architecture Diagram

3.3 Deployment Diagram

In Figure 3.5 we can see our deployment diagram. This diagram illustrates the physical deployment of our system components, including the servers, databases, and external services. Each component is represented as a node, and the relationships between them are depicted with lines connecting the nodes. The diagram also indicates the communication protocols used between components, such as Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS).

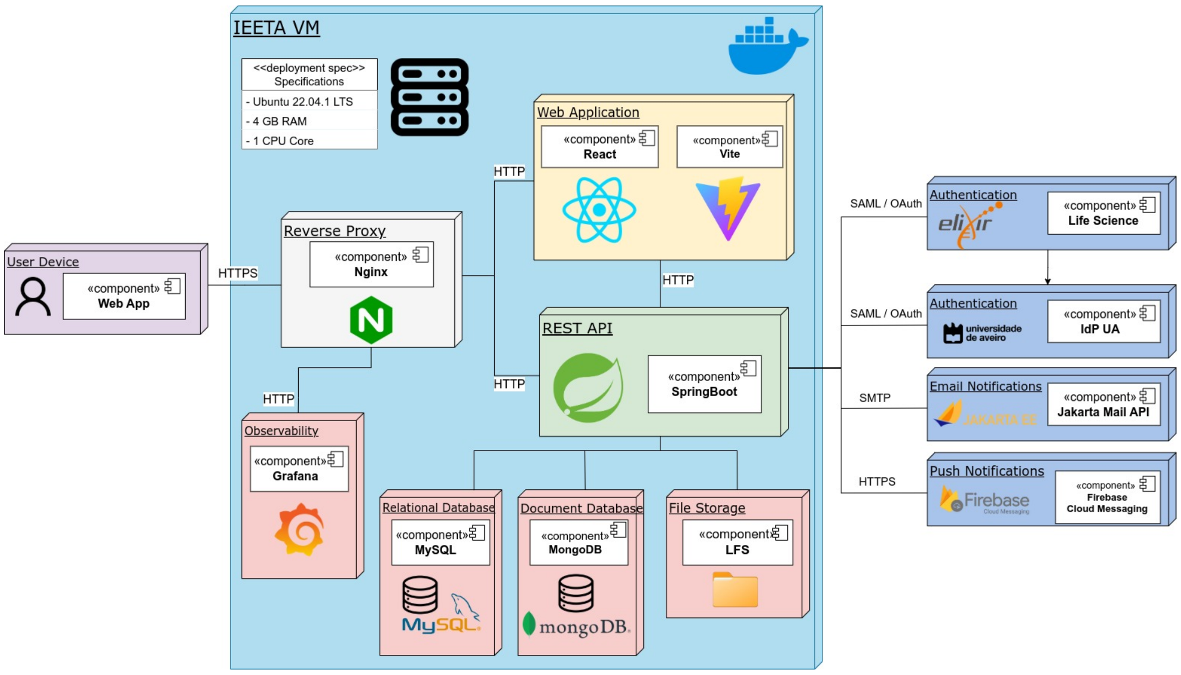


Figure 3.5: Deployment Diagram

3.3.1 Docker

Through Docker we separate all of our services into different containers, allowing them work independently, while also being able to communicate with each other.

frontend. The frontend container is responsible for serving the Web Application. It uses a *Dockerfile* to build the application and serve it using Nginx. The frontend container communicates with the backend services through HTTP requests.

backend. The backend container handles the core business logic of the application. It is built using a *Dockerfile* and an already compiled version of the backend. The backend container communicates with the frontend and the database containers.

mysql. Stores the relational data for the application. The data is persisted in a volume to ensure that it is not lost when the container is stopped or removed.

mongodb. Stores the MEC details in a flexible document-based format. Similar to the MySQL container, the data is persisted in a volume to ensure data integrity and persistence.

nginx. The Nginx container acts as a reverse proxy, routing incoming requests to the appropriate backend services, it listens on port 80 and 443. It also serves static files from the frontend container. The Nginx container is configured to handle HTTPS requests.

grafana. This container has the grafana image, which will allow us to access the system's information from the exterior. It brings a pre-defined dashboard and datasource with it, so that no setup is required.

loki. Loki will send the logs it captures to grafana, brought by promtail.

promtail. Promtail captures logs in the backend, and pushes those logs to Loki

3.3.2 Docker Compose

The Docker Compose is the orchestrator, it defines the services, networks, and volumes used in the deployment. It specifies the build context for each service, the ports to be exposed, and the environment variables required for each container. The Docker Compose file also defines the network configuration, allowing the containers to communicate with each other seamlessly. It also runs health checks to ensure that the services are running correctly and can restart them if necessary.

3.3.3 Production VM

Our production VM is hosted in the UA's infrastructure, specifically in **IEETA**. It has limited resources, with only 4GB of RAM and 1 CPU core, but this can be upgraded on demand, in a situation our system needs more processing power. As mentioned previously, only port 8080 is exposed.

We also registered the domain **evalmed.pt**, valid for 1 year, and the communications between the system and the client are done through HTTPS, which is possible thorugh certificates provided by ZeroSSL. The certificates were generated in the **Apache** proxy server sitting between the client and our reverse proxy, **Nginx**, not by us, but by the members responsible for said proxy.

CHAPTER 4

Implementation

4.1 Web Application

4.1.1 Architecture

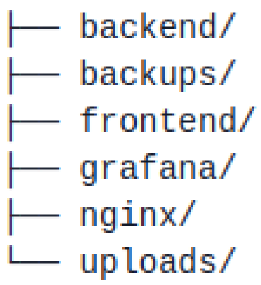
The implementation section addresses the mechanisms and methodologies employed in the development of the application. This includes a comprehensive overview of both the frontend and backend components. On the frontend side, we describe the structure of the user interface, the technologies used (such as frameworks or libraries), and how user interaction is handled. On the backend side, we detail the logic behind the core functionalities, including authentication mechanisms, data processing workflows, and data storage strategies.

We also provide an overview of the external libraries and dependencies that supported the development process, clearly specifying their respective roles and how they were integrated into the project. Security measures, including password hashing, token-based authentication, and data validation techniques, are addressed in detail. Furthermore, the communication between the frontend and backend, established through RESTful APIs, is thoroughly explained to demonstrate how data flows throughout the system. This section aims to provide a transparent and structured account of the implemented features, choices made during development, and the overall architecture of the application.

4.1.2 Organization

Our application is divided into many folders, each serving a specific service. Everything is stored in the folder *EvalMed* and we have the following structure:

- **frontend**: Frontend application, built with React and Vite.
- **backend**: Backend application, built with Spring Boot.
- **nginx**: Nginx configuration files for serving the frontend and backend applications.
- **uploads**: Folder for storing uploaded files, such as profile pictures.
- **grafana**: Grafana dashboard configuration files for monitoring and visualizing application metrics.
- **backups**: Folder for storing database backups.



```
├── backend/
├── backups/
├── frontend/
├── grafana/
├── nginx/
└── uploads/
```

Figure 4.1: Folder structure

4.1.3 Frontend

4.1.3.1 Frontend Folder Structure

The overall structure of the frontend directory is organized into several key folders. The `public` folder contains static assets such as the application's favicon and other publicly accessible files. The `node_modules` folder includes all external libraries and dependencies used in the frontend. These libraries support a wide range of functionalities, including icon rendering, Cascading Style Sheets (CSS) styling, data visualization (e.g., Chart.js), and PDF generation.

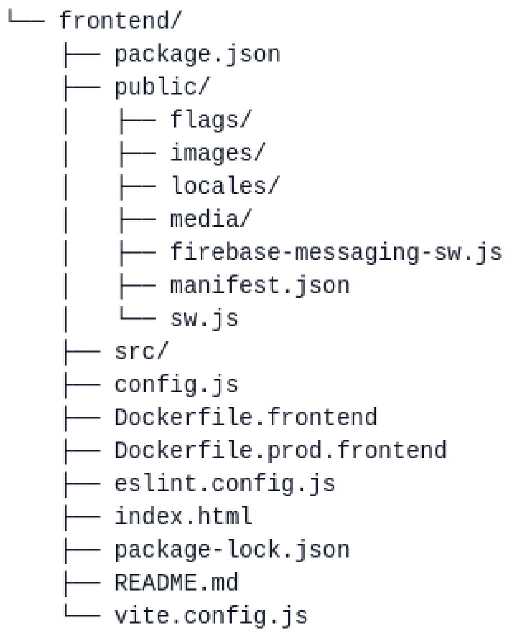
The locales folder on public contains all the JSON files responsible for the translation of hardcoded elements in the application, supporting multilingual functionality. Additionally, the media folder stores visual assets (e.g., pictures) that are used throughout the application interface.

4.1.3.2 Configuration Files

The project includes several important configuration files that play a critical role in the setup, communication, and functionality of the frontend application.

- **config.js**: Defines the base endpoint of the backend API (e.g., `/api/v1`), ensuring a centralized and consistent reference for API requests throughout the application.
- **vite.config.js**: Manages the build and development settings using Vite, a modern frontend build tool. This file includes configuration for plugin usage, module resolution, environment variables, and development server behavior.
- **firebase-messaging-sw.js**: A service worker file that enables FCM, allowing the application to receive push notifications even when running in the background or closed.
- **sw.js**: Acts as the main custom service worker. It performs several important tasks, including caching static assets, intercepting and handling API requests (especially POST requests to feedback endpoints), background synchronization using the `SyncManager API`, and offline fallback mechanisms. It also includes robust error logging and applies various security-related headers to improve resilience and user experience under unstable network conditions.
- **package.json** and **package-lock.json**: Define all third-party project dependencies and ensure that installations are consistent and reproducible across development en-

vironments. These files help maintain stability in the build process and avoid version conflicts.



```
└── frontend/
    ├── package.json
    ├── public/
    │   ├── flags/
    │   ├── images/
    │   ├── locales/
    │   ├── media/
    │   ├── firebase-messaging-sw.js
    │   ├── manifest.json
    │   └── sw.js
    ├── src/
    ├── config.js
    ├── Dockerfile.frontend
    ├── Dockerfile.prod.frontend
    ├── eslint.config.js
    ├── index.html
    ├── package-lock.json
    ├── README.md
    └── vite.config.js
```

Figure 4.2: Folder structure of the application's frontend

The `src` folder in our application is structured to ensure clarity, scalability, and ease of navigation. Each subfolder has a specific purpose, allowing developers to quickly understand and work within the project. Below is an overview of the main folders:

- **assets**: This folder contains static resources used throughout the application. For example, the custom font file (`.ttf`) used in the project is stored here.
- **components**: Contains all reusable components in the application. These are self-contained UI elements like buttons, inputs, or cards that can be used across multiple pages or features.
- **context**: Holds the React context providers that manage global state across the application. A key example is the user authentication context, which loads and shares the authenticated user data across components.
- **css**: This folder stores both global and modular CSS files that handle the styling of the entire application. We use `module.css` files specifically when styles are meant to apply only to a single component. This approach ensures style encapsulation, avoids conflicts between components, and promotes maintainable, modular code.
- **dtos** (Data Transfer Objects): Defines the data structures used to transfer and validate data between components and APIs. This ensures consistency and type safety across the application.
- **pages**: Contains the main route-level components of the application. Each subfolder typically represents a page and includes related logic, components, and styling specific to that page.

- **utils**: Includes utility functions and custom hooks that encapsulate reusable logic. For instance, the navigation hook that resets the scroll position to the top of the page when navigating between routes is located here, enhancing user experience.
- **App.jsx**: This is the root component of the application. It initializes the routing system and sets up global providers that manage context and shared state throughout the application. Key responsibilities of this file include:

- Wrapping the entire application with context providers such as `UserProvider`, `LayoutProvider`, `FeedbackProvider`, and `LoadingProvider`.
- Configuring the routing system using `react-router-dom`, defining both public and protected routes.
- Registering a service worker to enable offline capabilities and background sync (e.g., for feedback submission).
- Rendering the `OfflineWarning` component to notify users of offline status.
- Controlling route access with a `ProtectedLayout`, which checks user authentication before granting access to protected pages.

The routing structure is defined in a nested `AppRoutes` function. It includes:

- **Public Routes**: For pages such as login callbacks, terms of use, privacy policy, and the initial landing page.
- **Protected Routes**: For authenticated users only, including dashboard pages, evaluations, feedback flows, admin sections, and profile management.

The component checks whether authentication is complete before rendering protected routes. This prevents unauthorized access and avoids rendering sensitive content prematurely.

4.1.3.3 Service Workers

The application counts with 2 separate service workers, working in different scopes. They are visible for everyone, by clicking f12 on the browser and going to the "Application" tab.

FCM Service Worker

This service worker is responsible for listening to incoming push notifications coming from the Firebase Application we created.

Listing 4.1: Firebase Service Worker

```
importScripts('https://www.gstatic.com/firebasejs/9.2.0/firebase-app-compat.js');
importScripts('https://www.gstatic.com/firebasejs/9.2.0/firebase-messaging-compat.js');

firebase.initializeApp({
  apiKey: "AIzaSyDvv-JOgXACfYwSKCOPa7O12MC6subwuyM",
  authDomain: "evalmed.firebaseapp.com",
  projectId: "evalmed",
  storageBucket: "evalmed.firebasestorage.app",
  messagingSenderId: "282839473929",
  appId: "1:282839473929:web:d4673b01a984cfbf9ebdd8",
  measurementId: "G-HLTRM2WC26"
});

self.addEventListener('push', (event) => {
  const payload = event.data ? JSON.parse(event.data.text()) : {};
  const notificationTitle = payload.notification?.title || 'New
     Notification';
  const notificationOptions = {
    body: payload.notification?.body || 'You have a new message.',
    icon: payload.notification?.icon || './images/evalmed_144.png',
  };

  event.waitUntil(
    self.registration.showNotification(notificationTitle,
       notificationOptions)
  );
});
```

Note that the initialization data we can see on this service worker is all public, and is not expected to be a secret or hidden. This service worker waits for a notification coming from that Firebase Project, and once a push event is fired from that project, the notification is displayed even if the user is not in the website / app.

Main Service Worker

This Service Worker (SW) is the bread and butter of our offline functionalities. It serves many purposes:

- It stores failed POST requests due to internet connectivity issues in the IndexedDB, so that they can be retried when the user is back online
- It stores the static components of each page, and fetches them using an offline-first policy
- It stores responses from the API, and fetches them using a network-first policy

When the user enters the evalmed.pt domain, the SW is immediatly registered, already working on caching every HyperText Markup Language (HTML) page, for a total of roughly 20MB of cache storage. Fetching each HTML page becomes faster after that, as they are all cached. Additionally, the SW checks if there are any stored requests in the browser's IndexedDB, and if so, it attempts to send them to the backend, removing them from the queue after such. When the user attempts to make a POST request (regarding evaluations or feedback), if it fails due to internet connectivity issues, then the SW will store the request within the machine's IndexedDB. When the user attempts to fetch data from the backend when offline, then the SW will check if this endpoint was fetched previously, and if such, it returns the cached response instead of the backend's error data.

The SW's serve a primary role in both push notifications and offline functionality, both of which being core requirements set from the beginning, as this solution will be used frequently on the phone, and many times in places where internet connectivity is not guaranteed.

4.1.3.4 Prototype

We developed a low-fidelity prototype using Figma, where we designed a preliminary version of the application's interface. This prototype included all planned tasks and interactions, following the functional requirements analysis phase.

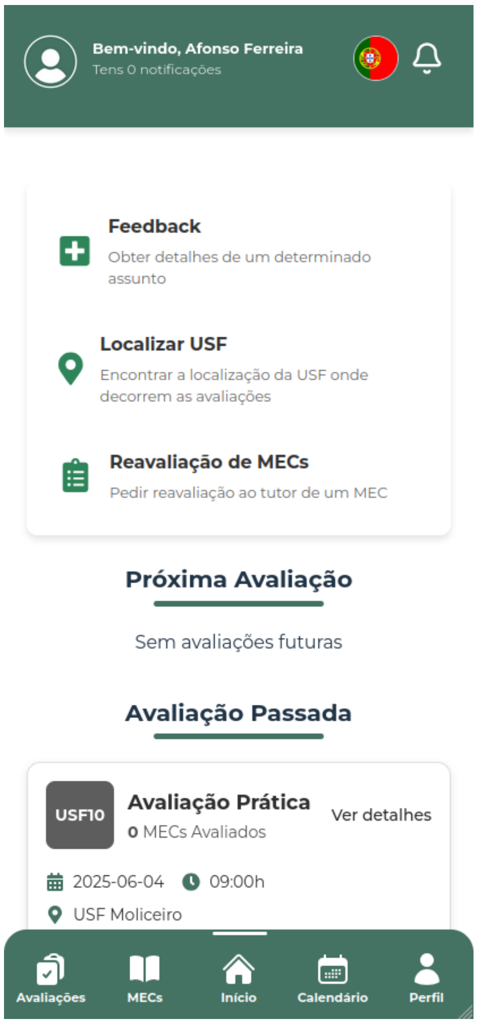
The prototype will be discussed in detail in the section on usability testing, where we evaluate how the application was tested by the target users for whom it was developed. These tests provided valuable feedback that informed the design and functionality of the final product.

4.1.3.5 Responsiveness and Cross-Device Compatibility

Our application was primarily designed with mobile devices in mind. However, considering the "Administrator" user type, who typically performs more complex operations, it was also important to ensure usability on desktop environments.

Although most of the application works well across devices by default, we relied heavily on the use of media queries to adapt custom elements - many of which are unique to our application and contribute to its differentiation from others currently available on the market.

. In the following figure, we present the appearance of the administrator's initial page on a mobile device. From this interface, the administrator can manage all the students enrolled in the course. It is possible to associate or dissociate tutors with students, and at the bottom of the page, a global statistical summary is displayed. This feature will be further explained in the section covering the main functionalities of the system.



Bem-vindo, Afonso Ferreira
Tens 0 notificações

Feedback
Obter detalhes de um determinado assunto

Localizar USF
Encontrar a localização da USF onde decorrem as avaliações

Reavaliação de MECs
Pedir reavaliação ao tutor de um MEC

Próxima Avaliação

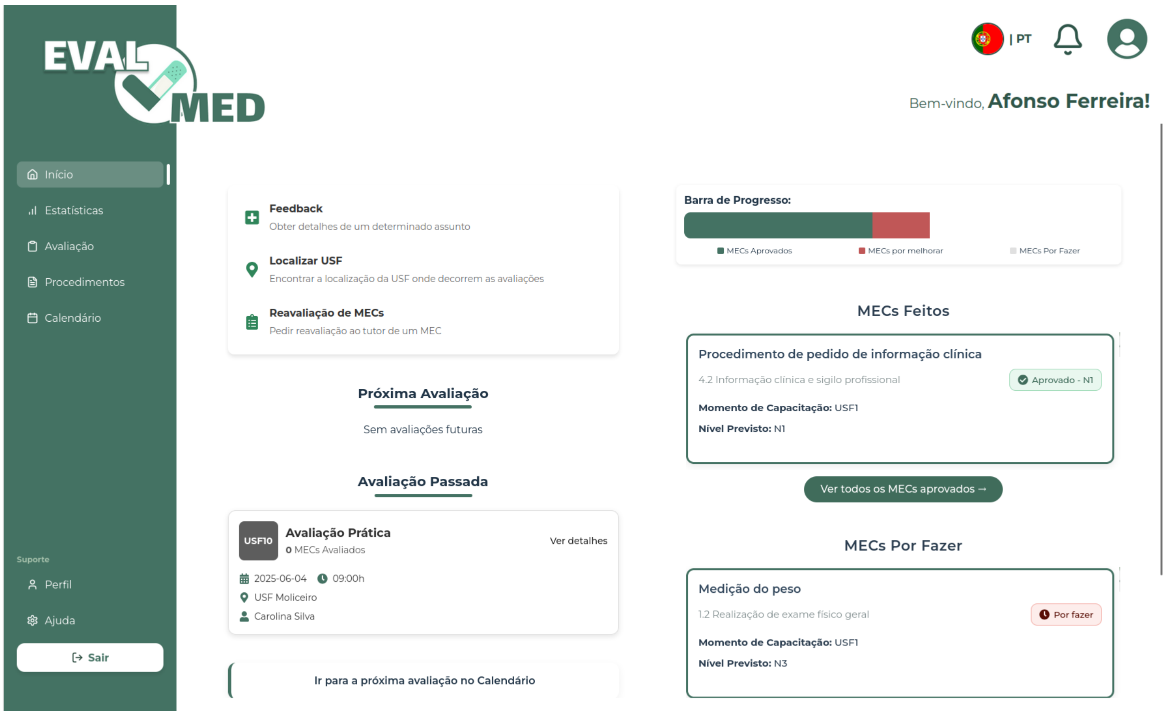
Sem avaliações futuras

Avaliação Passada

USF10 **Avaliação Prática** Ver detalhes
0 MECs Avaliados
2025-06-04 09:00h
USF Moliceiro

Figure 4.3: Responsive view of the student home page on a mobile device

. In the following figure, we show the tutor's initial page, which provides access to various functionalities. Through this interface, the tutor can easily navigate to all sections of the system. It enables efficient monitoring of students, ensuring that feedback is regularly provided and that all evaluations are progressing as expected.


EVAL MED

| PT

Bem-vindo, **Afonso Ferreira!**

Início
Estatísticas
Avaliação
Procedimentos
Calendário

Suporte
Perfil
Ajuda
Sair

Feedback
Obter detalhes de um determinado assunto

Localizar USF
Encontrar a localização da USF onde decorrem as avaliações

Reavaliação de MECs
Pedir reavaliação ao tutor de um MEC

Próxima Avaliação
Sem avaliações futuras

Avaliação Passada

USF10 **Avaliação Prática** Ver detalhes
0 MECs Avaliados
2025-06-04 09:00h
USF Moliceiro
Carolina Silva

Ir para a próxima avaliação no Calendário

Barra de Progresso:
MECs Aprovados · MECs por melhorar · MECs Por Fazer

MECs Feitos

Procedimento de pedido de informação clínica
4.2 Informação clínica e sigilo profissional — Aprovado - N1
Momento de Capacitação: USF1
Nível Previsto: N1

Ver todos os MECs aprovados →

MECs Por Fazer

Medição do peso
1.2 Realização de exame físico geral — Por fazer
Momento de Capacitação: USF1
Nível Previsto: N3

Figure 4.4: Student's main interface as displayed on a desktop device

4.1.3.6 Implemented Main Functionalities

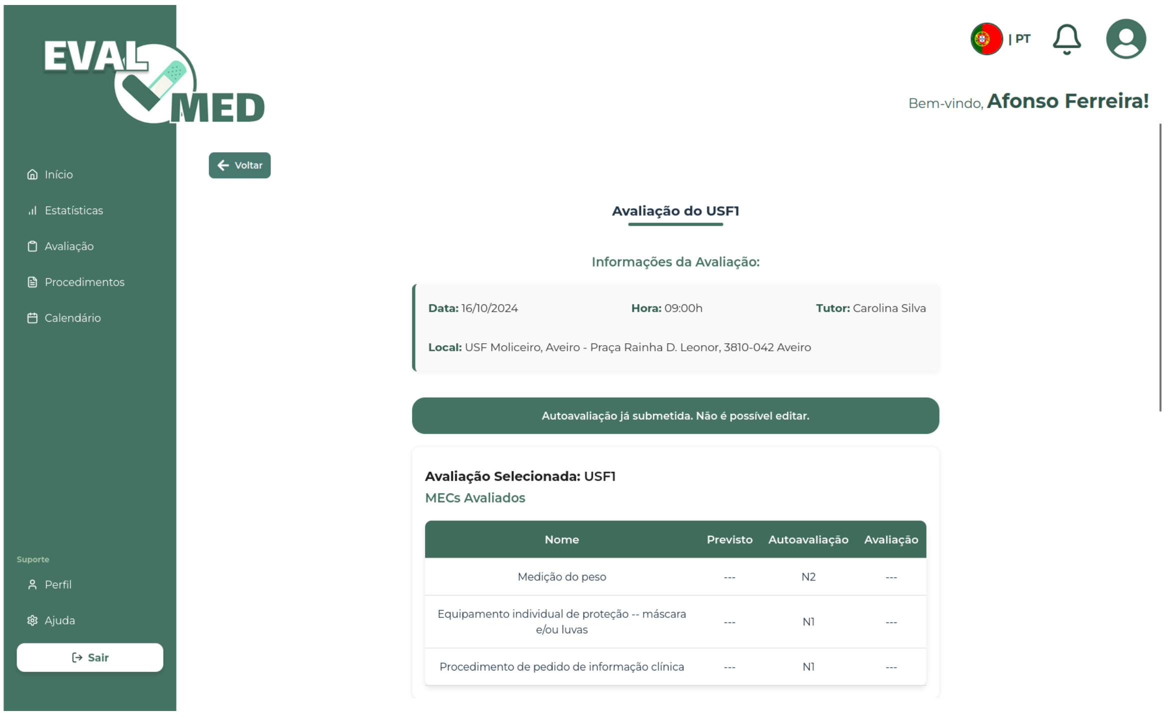
Our application features a highly intuitive interface, which received a notably high SUS score during usability testing, as detailed in Section 5. For the different user roles (Student, Tutor, and Administrator), we have developed a wide range of functionalities that align with the objectives defined by the stakeholders. These features not only meet the specified requirements but also demonstrate superior performance compared to other existing platforms.

We present the main functionalities implemented for each user type, along with corresponding interface images to provide a clear and visual representation of the user experience within the system.

4.1.3.7 Student

- Perform self-evaluation

Students will undergo several evaluations throughout the academic year, each covering a set of MECs. For each evaluation, students are required to submit a self-assessment within a 7-day period prior to the evaluation date. This self-assessment must be completed in advance so that the tutor can understand how the student perceives their practical readiness regarding the competencies to be assessed.



Data: 16/10/2024 **Hora:** 09:00h **Tutor:** Carolina Silva

Local: USF Moliceiro, Aveiro - Praça Rainha D. Leonor, 3810-042 Aveiro

Autoavaliação já submetida. Não é possível editar.

Avaliação Selecionada: USF1

MECs Avaliados

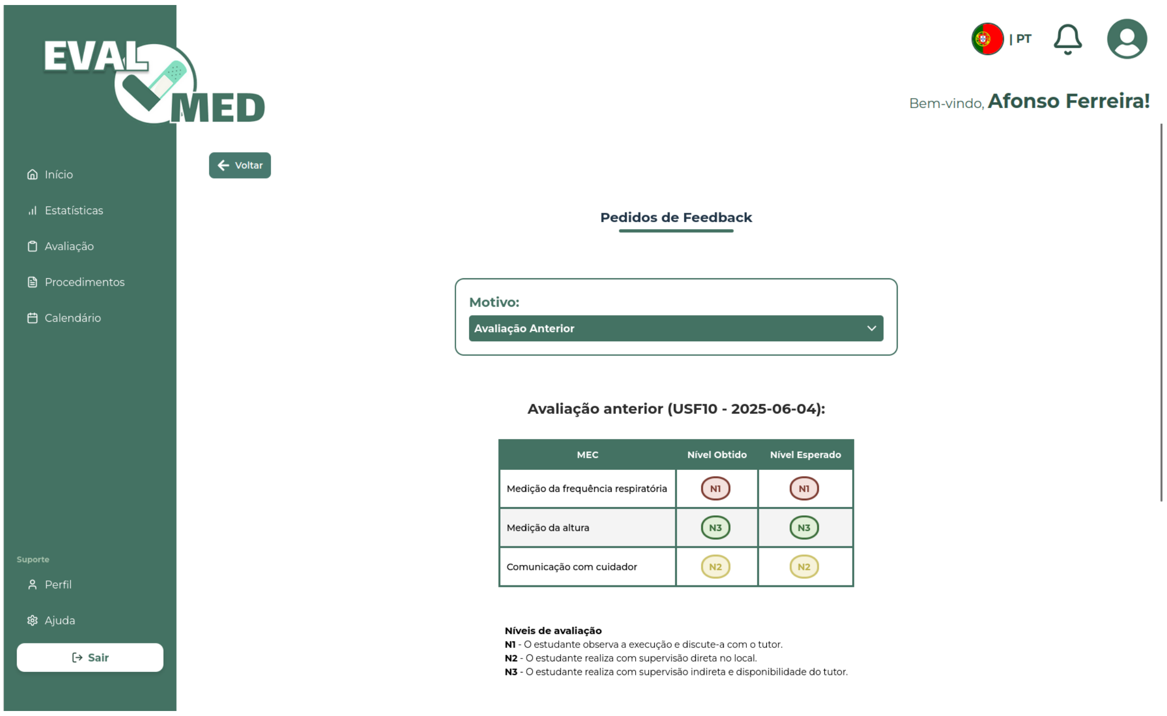
Nome	Previsto	Autoavaliação	Avaliação
Medição do peso	---	N2	---
Equipamento individual de proteção -- máscara e/ou luvas	---	N1	---
Procedimento de pedido de informação clínica	---	N1	---

Figure 4.5: Page of a specific evaluation where the student completes the self-assessment

A dedicated page has been developed where students can carry out their self-assessment. Once submitted, the self-assessment can be modified later if the student believes their performance expectations have changed. On this page, students can see which MECs will be assessed, along with the expected performance level for each. They can also view details related to the date, time, and location of the evaluation.

- Submit feedback

To submit feedback, the student must select a reason, with the available options being: feedback regarding the last evaluation, the upcoming evaluation, or a specific MEC (Minimum Expected Competency) that will be assessed. This functionality allows students to maintain ongoing communication with their tutor. In this way, they remain informed about their strengths and weaknesses in practice and can also raise any questions they may have - whether theoretical or practical - about a specific MEC.



Pedidos de Feedback

Motivo: Avaliação Anterior

Avaliação anterior (USF10 - 2025-06-04):

MEC	Nível Obtido	Nível Esperado
Medição da frequência respiratória	N1	N1
Medição da altura	N3	N3
Comunicação com cuidador	N2	N2

Níveis de avaliação
N1 - O estudante observa a execução e discute-a com o tutor.
N2 - O estudante realiza com supervisão direta no local.
N3 - O estudante realiza com supervisão indireta e disponibilidade do tutor.

Figure 4.6: Student – Feedback submission interface

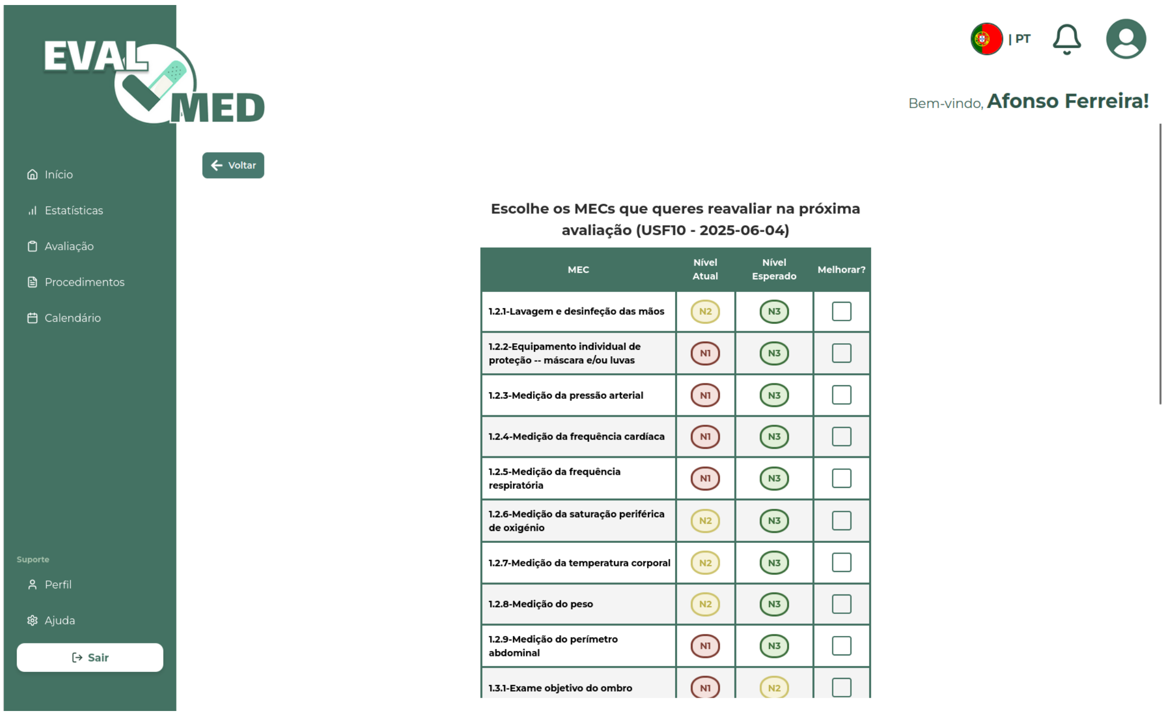
The interface includes a dropdown menu with the aforementioned options, where the student selects the reason for the feedback. If "specific MEC" is selected, a list of all MECs included in the student's evaluation is displayed. The student must then enter their message in a text box, either by typing or using the voice-to-text functionality.

Once submitted, the student awaits a response from their tutor.

- Request a reevaluation

After completing a practical evaluation composed of MECs, if a student does not achieve the expected performance level required for full completion of a MEC, they have the option to submit a *Possible Improvement* request. This allows the uncompleted MECs to be automatically transferred to the next evaluation, giving students the opportunity to retake them and aim to meet the required performance level. This reevaluation request must be approved by the tutor and should only be proposed if the student believes they can improve and the tutor agrees that the student is prepared for further practice.

The tutor has access to specific interfaces to review and accept reevaluation requests.



Escolhe os MECs que queres reavaliar na próxima avaliação (USF10 - 2025-06-04)

MEC	Nível Atual	Nível Esperado	Melhorar?
1.2.1-Lavagem e desinfeção das mãos	N2	N3	☐
1.2.2-Equipamento individual de proteção -- máscara e/ou luvas	N1	N3	☐
1.2.3-Medição da pressão arterial	N1	N3	☐
1.2.4-Medição da frequência cardíaca	N1	N3	☐
1.2.5-Medição da frequência respiratória	N1	N3	☐
1.2.6-Medição da saturação periférica de oxigénio	N2	N3	☐
1.2.7-Medição da temperatura corporal	N2	N3	☐
1.2.8-Medição do peso	N2	N3	☐
1.2.9-Medição do perímetro abdominal	N1	N3	☐
1.3.1-Exame objetivo do ombro	N1	N2	☐

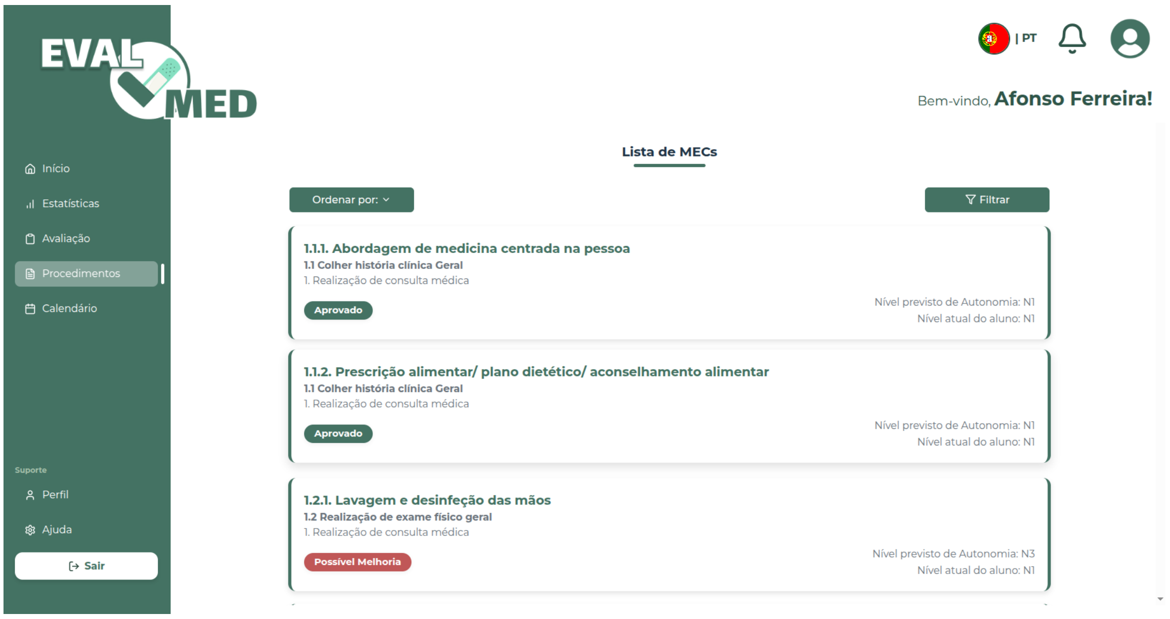
Figure 4.7: Student – Reevaluation request form

The reevaluation request page displays a table listing all MECs that the student has not yet fully completed. The student can select the MECs they believe they are ready to improve upon. This request will then be reviewed by the tutor through their dedicated interface.

After submission, the student can also leave an optional message to the tutor to provide any relevant information regarding the reevaluation request. If the student has any doubts about a specific MEC, they can click on it to be redirected to the MEC specification page. This feature ensures that students are well informed before selecting the MECs they wish to be reevaluated on, promoting more thoughtful and prepared decision-making.

- View a list of all the procedure MECs

Students can access a list of all the procedures planned for the specific curricular year. This list includes cards for each MEC, which contain essential information such as the MEC's name, current autonomy level and expected autonomy level. Besides that, the students also have access to a filtering option where they can filter the MECs by their current autonomy level, expected autonomy level, competency, the EPA they are part of and others. Since there are about 30 MECs per curricular year, this filtering option is very useful for students to quickly find the MECs they are looking for.



Bem-vindo, **Afonso Ferreira!**

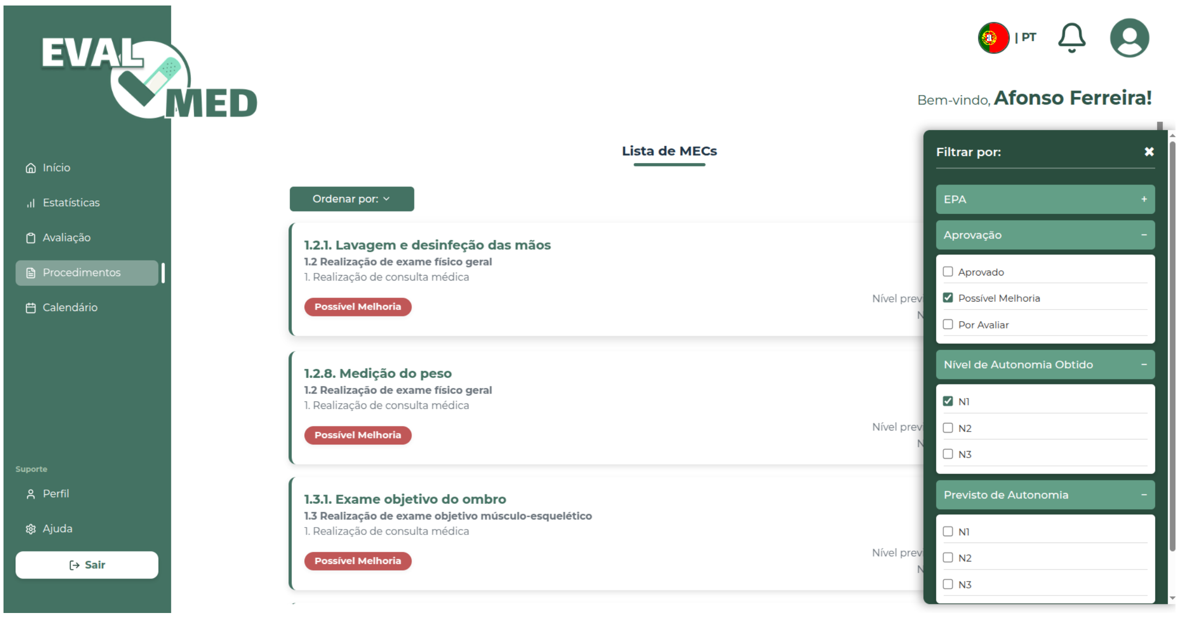
Lista de MECs

1.1.1. Abordagem de medicina centrada na pessoa
1.1 Colher história clínica Geral
1. Realização de consulta médica
Aprovado
Nível previsto de Autonomia: N1
Nível atual do aluno: N1

1.1.2. Prescrição alimentar/ plano dietético/ aconselhamento alimentar
1.1 Colher história clínica Geral
1. Realização de consulta médica
Aprovado
Nível previsto de Autonomia: N1
Nível atual do aluno: N1

1.2.1. Lavagem e desinfeção das mãos
1.2 Realização de exame físico geral
1. Realização de consulta médica
Possível Melhoria
Nível previsto de Autonomia: N3
Nível atual do aluno: N1

Figure 4.8: Student – List of all MECs



Bem-vindo, **Afonso Ferreira!**

Lista de MECs

1.2.1. Lavagem e desinfeção das mãos
1.2 Realização de exame físico geral
1. Realização de consulta médica
Possível Melhoria
Nível prev

1.2.8. Medição do peso
1.2 Realização de exame físico geral
1. Realização de consulta médica
Possível Melhoria
Nível prev

1.3.1. Exame objetivo do ombro
1.3 Realização de exame objetivo músculo-esquelético
1. Realização de consulta médica
Possível Melhoria
Nível prev

Filtrar por:
EPA +
Aprovação −
☐ Aprovado
☑ Possível Melhoria
☐ Por Avaliar
Nível de Autonomia Obtido −
☑ N1
☐ N2
☐ N3
Previsto de Autonomia −
☐ N1
☐ N2
☐ N3

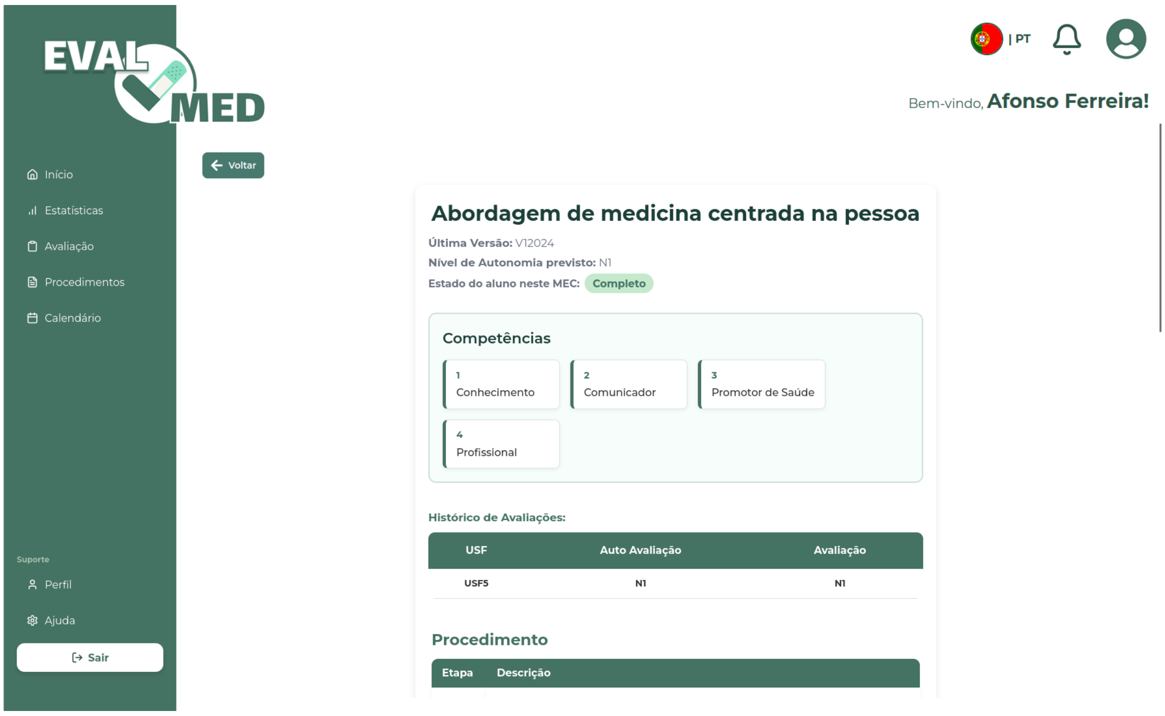
Figure 4.9: Student – Filtering MECs by 'Approval Status' and 'Current Autonomy Level'

- View a specific procedure MEC

Students have the ability to search for any information related to MECs under the "Procedures" tab. This section provides comprehensive details about each MEC, including its associated competencies (small key concepts that make up each MEC and are progressively acquired by students), the expected performance level, the level achieved, and the self-assessed level submitted by the student. Additionally, students can view all the steps involved in completing each MEC and the procedures associated with its execution.

They also have access to instructional materials uploaded by the tutor, such as video resources and bibliographic references, which provide further theoretical and practical context for the MECs.

Finally, there is a feature that allows students to download a structured PDF version of the procedure. This is particularly useful for students who prefer not to read directly from the platform or who need a printed version of the MEC procedure. The downloadable document includes all the information presented in the interface in an organized format.



Abordagem de medicina centrada na pessoa

Última Versão: V12024
Nível de Autonomia previsto: N1
Estado do aluno neste MEC: Completo

Competências

1 Conhecimento
2 Comunicador
3 Promotor de Saúde
4 Profissional

Histórico de Avaliações:

USF	Auto Avaliação	Avaliação
USF5	N1	N1

Procedimento

Etapa	Descrição

Figure 4.10: Student – Procedure view page

In this section, the student must select the specific MEC they wish to consult. Upon selection, all related information becomes available in a clear and organized layout. This includes a breakdown of the required steps, detailed procedural guidelines, and any associated competencies.

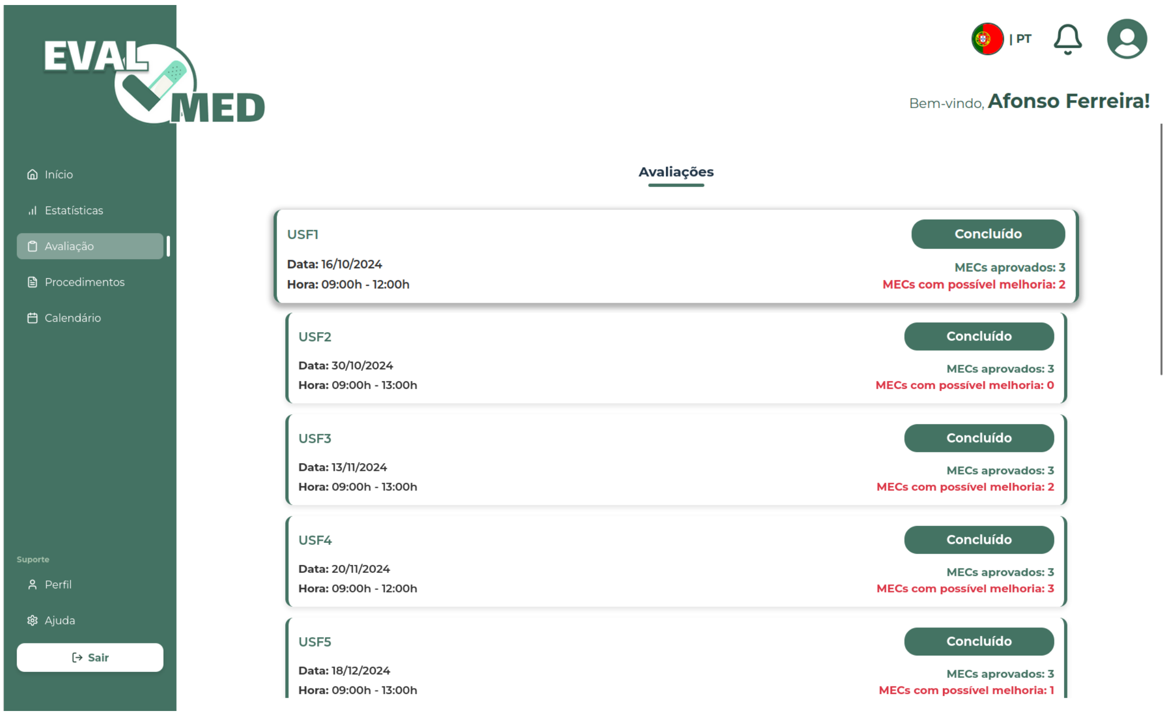
This centralized access not only promotes student autonomy but also ensures they are better prepared for both self-assessment and practical performance. By understanding the expectations and methodology associated with each MEC, students can plan and structure their learning process more effectively.

Furthermore, the combination of multimedia resources, bibliographic references, and the ability to export a detailed PDF enables students to engage with the material in multiple formats. Whether they are studying independently, revising for an evaluation, or preparing for a reevaluation, the "Procedures" tab serves as a comprehensive and flexible reference point.

Overall, this functionality reinforces transparency in the evaluation process and encourages proactive learning, enabling students to take greater responsibility for their progress and practical development.

- View all evaluations: completed, pending, or flagged for improvement

On this page, the student has access to all available evaluations. From here, they can later complete the self-assessment and consult the evaluation details, as previously described. The interface provides only summary information for each evaluation, such as the location, time, and evaluation number.



USF1 Concluído
Data: 16/10/2024
Hora: 09:00h - 12:00h
MECs aprovados: 3
MECs com possível melhoria: 2

USF2 Concluído
Data: 30/10/2024
Hora: 09:00h - 13:00h
MECs aprovados: 3
MECs com possível melhoria: 0

USF3 Concluído
Data: 13/11/2024
Hora: 09:00h - 13:00h
MECs aprovados: 3
MECs com possível melhoria: 2

USF4 Concluído
Data: 20/11/2024
Hora: 09:00h - 12:00h
MECs aprovados: 3
MECs com possível melhoria: 3

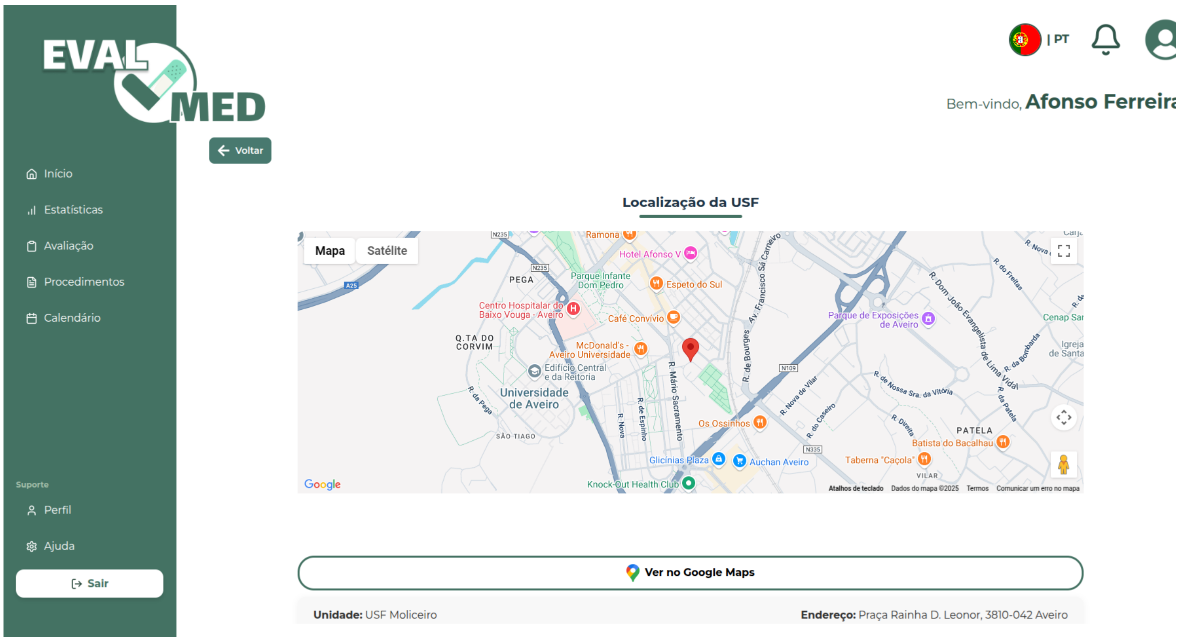
USF5 Concluído
Data: 18/12/2024
Hora: 09:00h - 13:00h
MECs aprovados: 3
MECs com possível melhoria: 1

Figure 4.11: Student – Evaluations list

The page presents a list of all evaluations, including those that have already been completed, those marked for possible improvement, and those still pending. This overview helps students keep track of their evaluation history and upcoming assessments in an organized and accessible way.

- View the location of their evaluations (USFs)

Both students and tutors have access to a location assistance feature in case they need guidance to reach their assigned USF. This functionality was first implemented using the `react-leaflet` library for map rendering and to fetch for the location of the USF. However, the API had limitations in finding the certain location of the USF and when it did find the location , it was not accurate enough to be useful. Therefore, we decided to use the Google Maps API, which provides a more reliable and accurate location service.



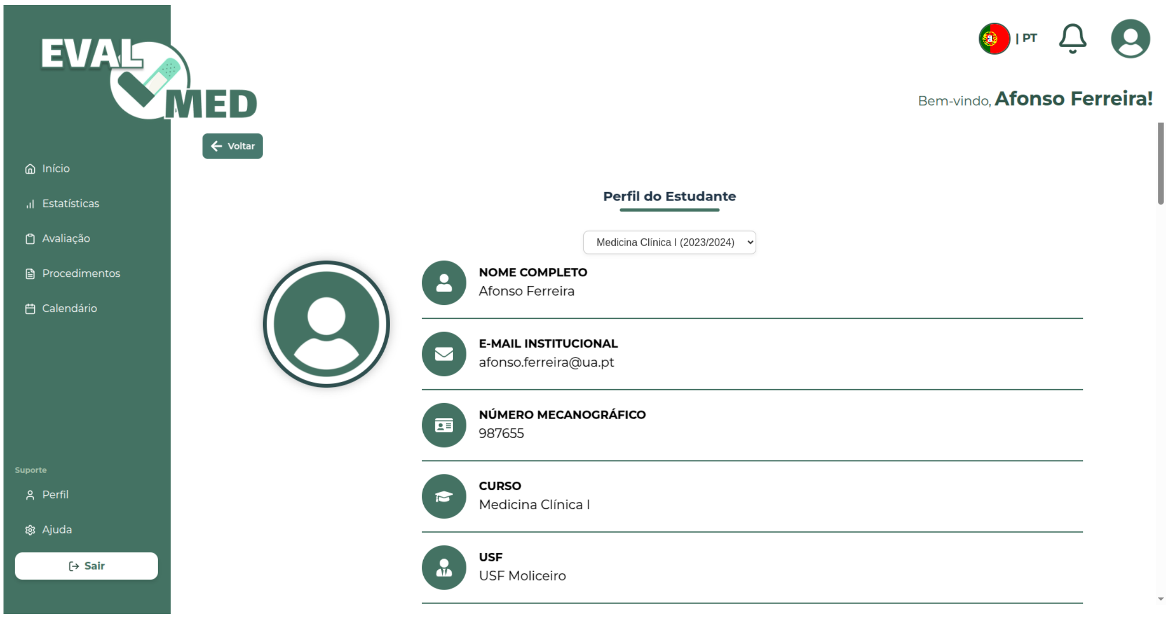
Unidade: USF Moliceiro

Endereço: Praça Rainha D. Leonor, 3810-042 Aveiro

Figure 4.12: Student – Location feature

- View past curricular year details

On this page, which is accessible from the student's profile, the student can view details from past curricular years. This includes information about the number of MECs that were completed, the time of each evaluation and the number of MECs that were planned for each evaluation and the feedback given by the tutor during that curricular year. This feature allows students to reflect on their past performance and track their progress over time and go back to any content they might have forgotten and want to review again. In the top of the page there's a button where the users can select the curricular year they want to view, and the page will update accordingly to show the relevant information.



Bem-vindo, **Afonso Ferreira!**

Perfil do Estudante

Medicina Clínica I (2023/2024)

NOME COMPLETO
Afonso Ferreira

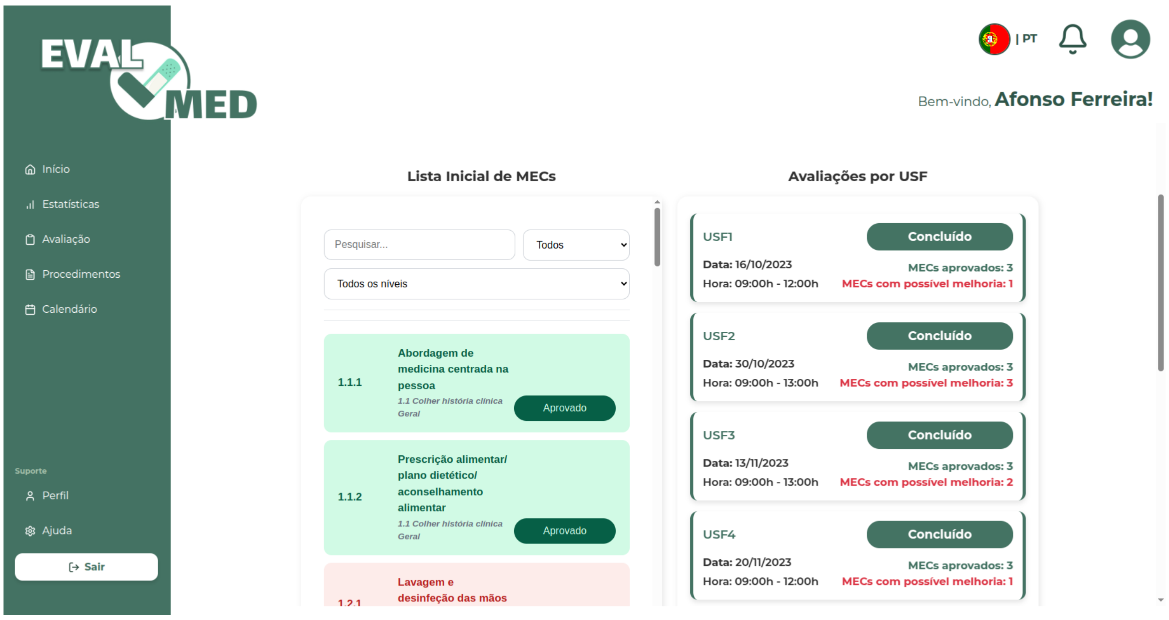
E-MAIL INSTITUCIONAL
afonso.ferreira@ua.pt

NÚMERO MECANOGRÁFICO
987655

CURSO
Medicina Clínica I

USF
USF Moliceiro

Figure 4.13: Student – Past curricular year details (Profile Information)



Bem-vindo, **Afonso Ferreira!**

Lista Inicial de MECs

1.1.1 **Abordagem de medicina centrada na pessoa** *1.1 Colher história clínica Geral* Aprovado

1.1.2 **Prescrição alimentar/ plano dietético/ aconselhamento alimentar** *1.1 Colher história clínica Geral* Aprovado

1.2.1 **Lavagem e desinfeção das mãos**

Avaliações por USF

USF1 — Concluído
Data: 16/10/2023
Hora: 09:00h - 12:00h
MECs aprovados: 3
MECs com possível melhoria: 1

USF2 — Concluído
Data: 30/10/2023
Hora: 09:00h - 13:00h
MECs aprovados: 3
MECs com possível melhoria: 3

USF3 — Concluído
Data: 13/11/2023
Hora: 09:00h - 13:00h
MECs aprovados: 3
MECs com possível melhoria: 2

USF4 — Concluído
Data: 20/11/2023
Hora: 09:00h - 13:00h
MECs aprovados: 3
MECs com possível melhoria: 1

Figure 4.14: Student – Past curricular year details (MECs and Evaluations)

4.1.3.8 Tutor

- Change between students

Tutors can have multiple students assigned to them, and they can easily switch between these students using a dropdown menu located in the side bar in desktop view or in the top NavBar in mobile view. By switching to another student, the page reloads with the new student's information, allowing the tutor to access all functionalities related to that specific student in any page of the application.

- Perform evaluations

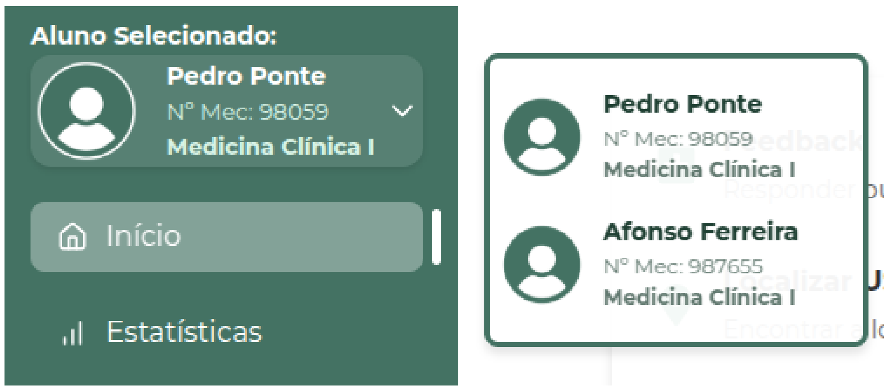
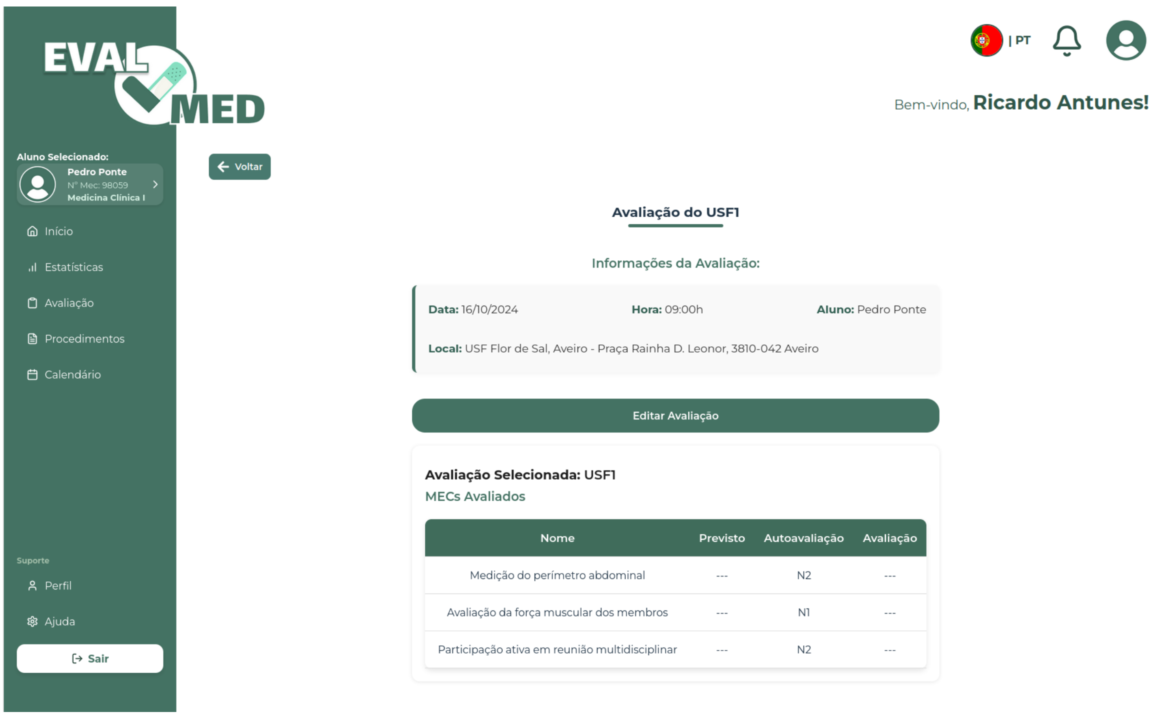


Figure 4.15: Tutor – Changing between students

Just like the student, the tutor can access each evaluation composed of the procedures introduced by the administrator, whose interface will be presented later in their functionalities. In each evaluation, the tutor must check if the student has already completed the self evaluation, however, if it has not been done yet, the tutor can still proceed with the evaluation. The tutor must conduct the evaluation during the student's practical assessment and submit it within one hour after the proposed end time for the evaluation. It is also possible for the tutor to leave feedback related to their performance, highlighting the strengths and weaknesses they identified, so that the student can improve their competencies. If the tutor is unsure about what to write, they have the option to generate pre-written phrases from a JSON file, making this process more efficient.

The tutor has the ability to plan an evaluation by selecting the MECs that will be part of the student's assessment.



Avaliação do USF1

Informações da Avaliação:

Data: 16/10/2024 **Hora:** 09:00h **Aluno:** Pedro Ponte

Local: USF Flor de Sal, Aveiro - Praça Rainha D. Leonor, 3810-042 Aveiro

Avaliação Selecionada: USF1

MECs Avaliados

Nome	Previsto	Autoavaliação	Avaliação
Medição do perímetro abdominal	---	N2	---
Avaliação da força muscular dos membros	---	N1	---
Participação ativa em reunião multidisciplinar	---	N2	---

Figure 4.16: Tutor – Performing an evaluation

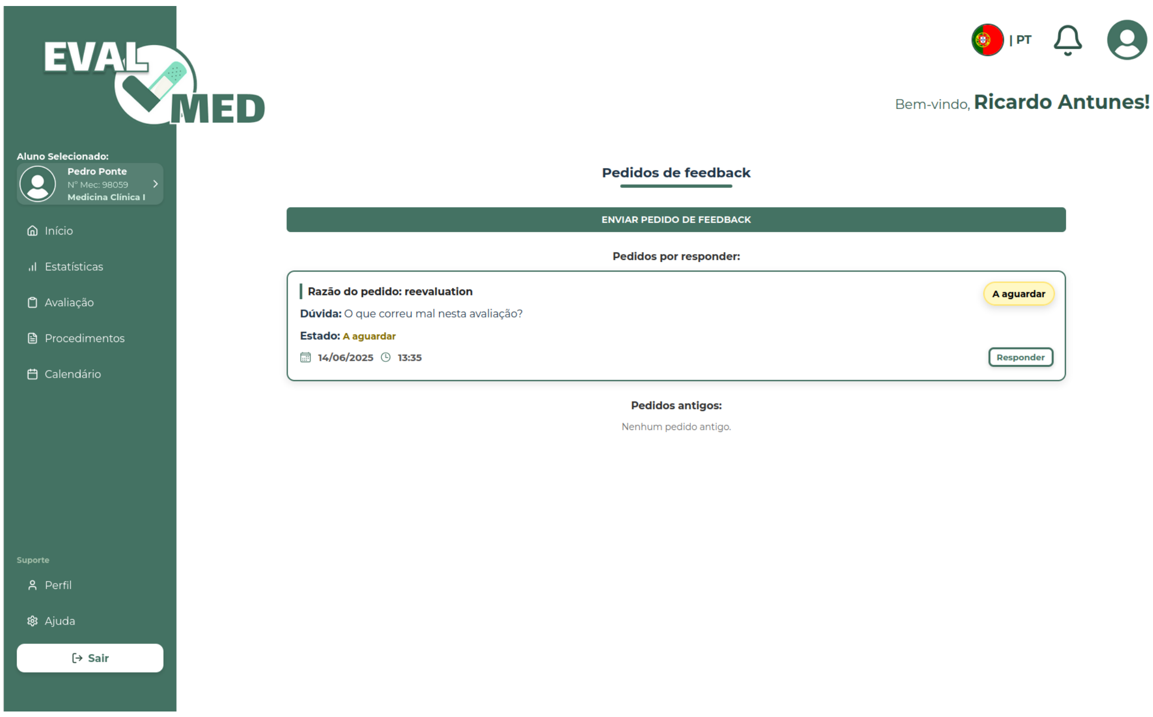
The page displays all relevant information regarding a given evaluation. It includes a button that allows the tutor to either add a new evaluation or edit an existing one that has already been conducted.

Additionally, a table is presented showing all evaluations the student has undergone, including the self-assessment submitted by the student and, if already provided, the tutor's assessment.

At the bottom of the page, there are text boxes containing the tutor's comments and feedback, offering qualitative insights into the student's performance.

- Submit and respond to feedback

The tutor has the ability to review all student feedback submissions, whether pending or already addressed. It is the tutor's responsibility to respond to any feedback that has been requested by the student. Additionally, the tutor can initiate communication by requesting feedback from the student, selecting a reason in the same way the student does.



Bem-vindo, **Ricardo Antunes!**

Pedro Ponte
Nº Mec: 98059
Medicina Clínica I

Pedidos de feedback

Pedidos por responder:

Razão do pedido: reevaluation A aguardar
Dúvida: O que correu mal nesta avaliação?
Estado: A aguardar
14/06/2025 13:35

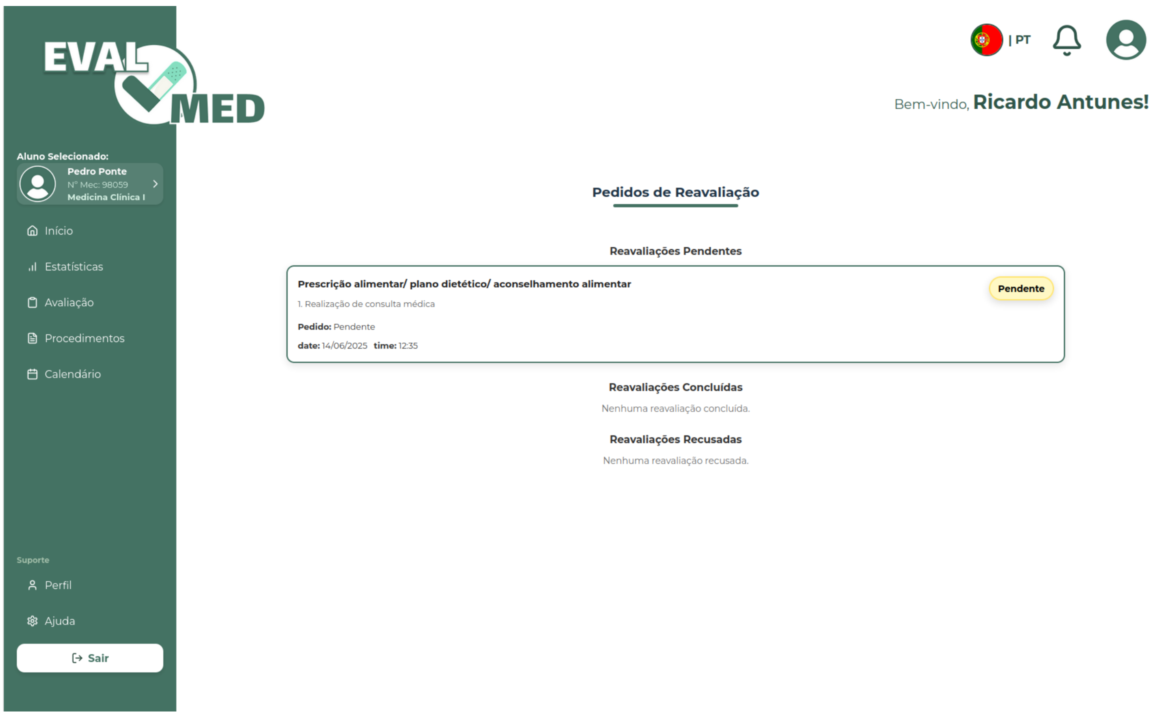
Pedidos antigos:
Nenhum pedido antigo.

Figure 4.17: Tutor – Feedback submission and response interface

The interface provides a "Respond" option for any feedback appearing in the pending section. Tutors can also request feedback from students, choosing from the same predefined reasons: last evaluation, upcoming evaluation, or a specific MEC. This is particularly useful when the tutor identifies areas where the student may be struggling in the practical execution of a given MEC.

- Respond to reevaluation requests

As with the feedback system, the tutor must be able to view and manage all reevaluation requests submitted by students.



Bem-vindo, Ricardo Antunes!

Aluno Selecionado:
Pedro Ponte
Nº Mec: 98059
Medicina Clínica I

Pedidos de Reavaliação

Reavaliações Pendentes

Prescrição alimentar/ plano dietético/ aconselhamento alimentar
Pendente
1. Realização de consulta médica
Pedido: Pendente
date: 14/06/2025 time: 12:35

Reavaliações Concluídas
Nenhuma reavaliação concluída.

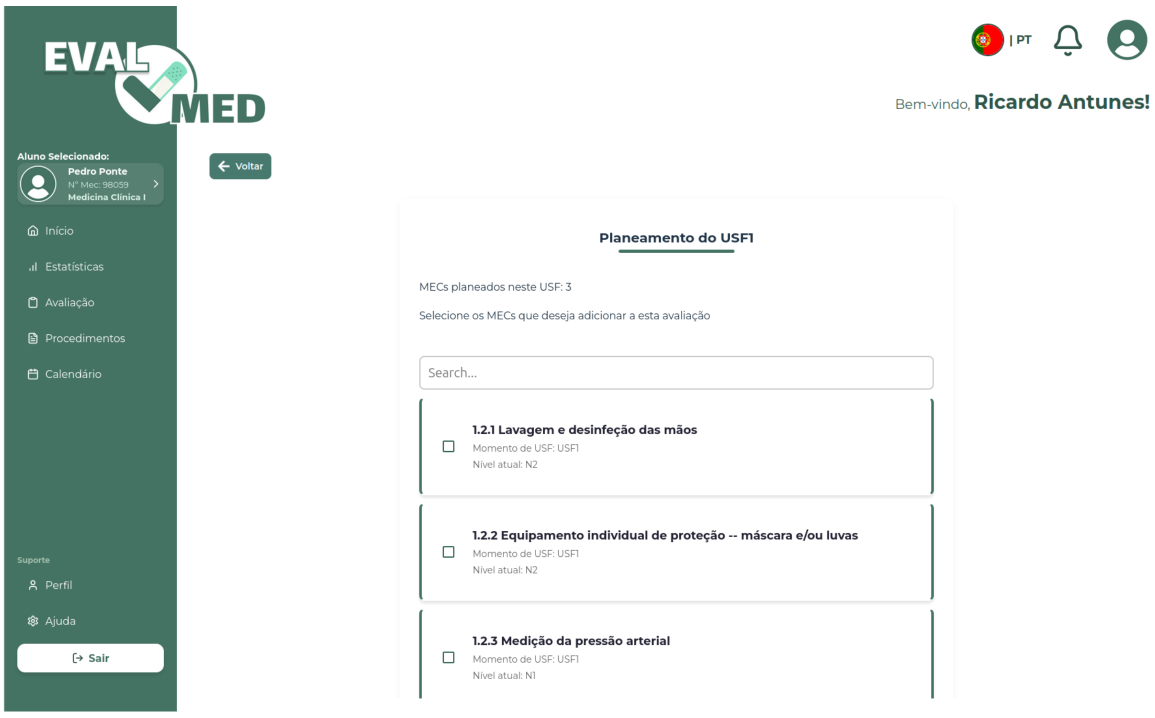
Reavaliações Recusadas
Nenhuma reavaliação recusada.

Figure 4.18: Tutor – Reevaluation response screen

For each reevaluation request, the tutor is presented with the options to either accept or cancel the request, based on their judgment and the details provided by the student, including any accompanying feedback. This process ensures that reevaluations are approved only when appropriate, supporting a pedagogically sound and individualized learning path.

- Plan an evaluation select the MECs

The tutor has the ability to select the MECs that each student will be evaluated on, following the guidelines established by the course coordinators. The selected MECs will subsequently become visible to the student through their interface.



Planeamento do USF1

MECs planeados neste USF: 3

Selecione os MECs que deseja adicionar a esta avaliação

Search...

- [ ] **1.2.1 Lavagem e desinfeção das mãos**
  Momento de USF: USF1
  Nível atual: N2
- [ ] **1.2.2 Equipamento individual de proteção -- máscara e/ou luvas**
  Momento de USF: USF1
  Nível atual: N2
- [ ] **1.2.3 Medição da pressão arterial**
  Momento de USF: USF1
  Nível atual: N1

Figure 4.19: Tutor – Evaluation planning interface

The interface displays a set of MECs that must be completed by the student by the end of the academic year. The tutor can mark the required MECs by selecting the corresponding checkboxes. Only MECs that haven't been evaluated or that the student hasn't reached the necessary autonomy level can be added to the evaluation. This functionality provides tutors with flexibility while ensuring that each student's evaluation path remains aligned with the course structure.

4.1.3.9 Administrator

The administrator, as the course instructor, plays a key role in ensuring that both tutors and students have access to all necessary information. This includes enabling tutors to access their assigned students, providing them with the necessary MECs and updated procedures.

- See older curricular years

Almost every page in the administrator interface includes an Year Selector, allowing the administrator to view information from previous curricular years. If the administrator selects a previous year, the interface will update and show the relevant information for that specific year. It's also used in the 'Program UC' page, where the administrator can change to next curricular year and plan the new curricular year by adding new evaluations.

- View global statistics

The administrator must be able to view up-to-date global statistics for all students enrolled in the course. This includes clear information such as the number of students with completed MECs, those still pending, and those marked for possible improvement. The administrator

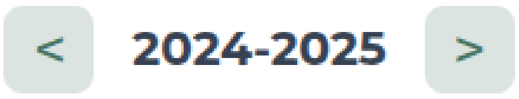


Figure 4.20: Administrator – Year Selector

also has access to the total number of reevaluations requested, feedback messages submitted, and the average student performance in the course.

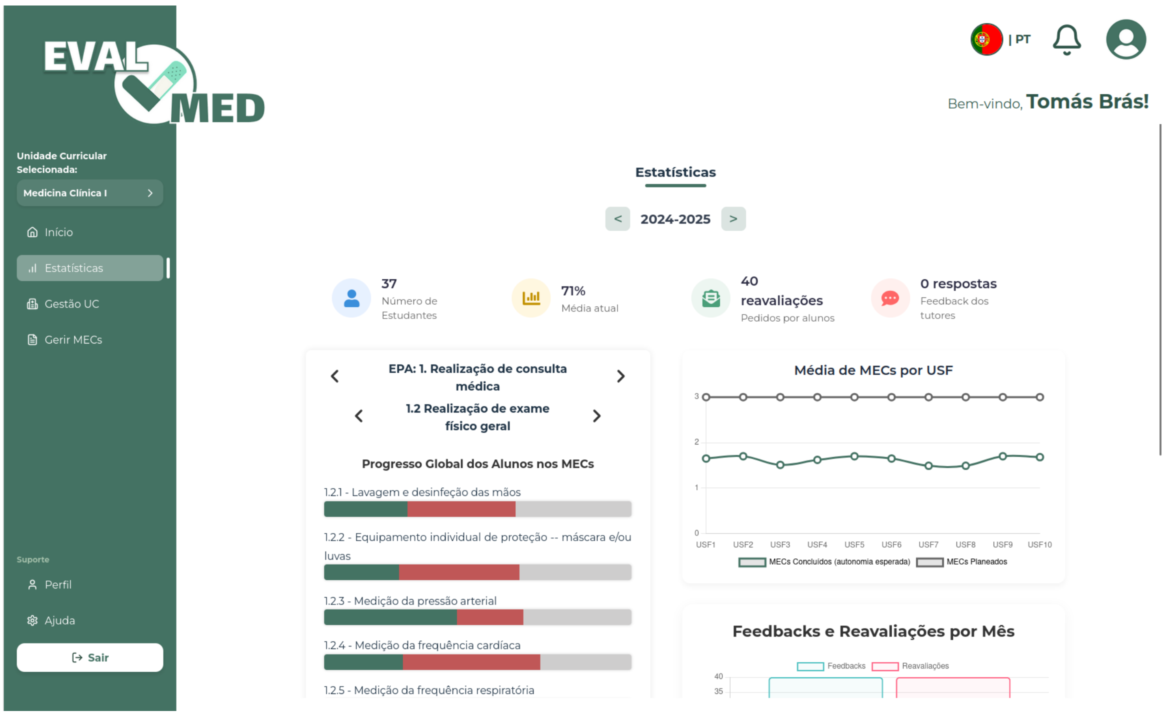

EVAL MED
| PT
Bem-vindo, Tomás Brás!
Unidade Curricular Selecionada:
Medicina Clínica I
Início
Estatísticas
Gestão UC
Gerir MECs
Suporte
Perfil
Ajuda
Sair
Estatísticas
2024-2025
37 Número de Estudantes
71% Média atual
40 reavaliações Pedidos por alunos
0 respostas Feedback dos tutores
EPA: 1. Realização de consulta médica
1.2 Realização de exame físico geral
Progresso Global dos Alunos nos MECs
1.2.1 - Lavagem e desinfeção das mãos
1.2.2 - Equipamento individual de proteção -- máscara e/ou luvas
1.2.3 - Medição da pressão arterial
1.2.4 - Medição da frequência cardíaca
1.2.5 - Medição da frequência respiratória
Média de MECs por USF
USF1
USF2
USF3
USF4
USF5
USF6
USF7
USF8
USF9
USF10
MECs Concluídos (autonomia esperada)
MECs Planeados
Feedbacks e Reavaliações por Mês
Feedbacks
Reavaliações

Figure 4.21: Administrator – Global statistics dashboard

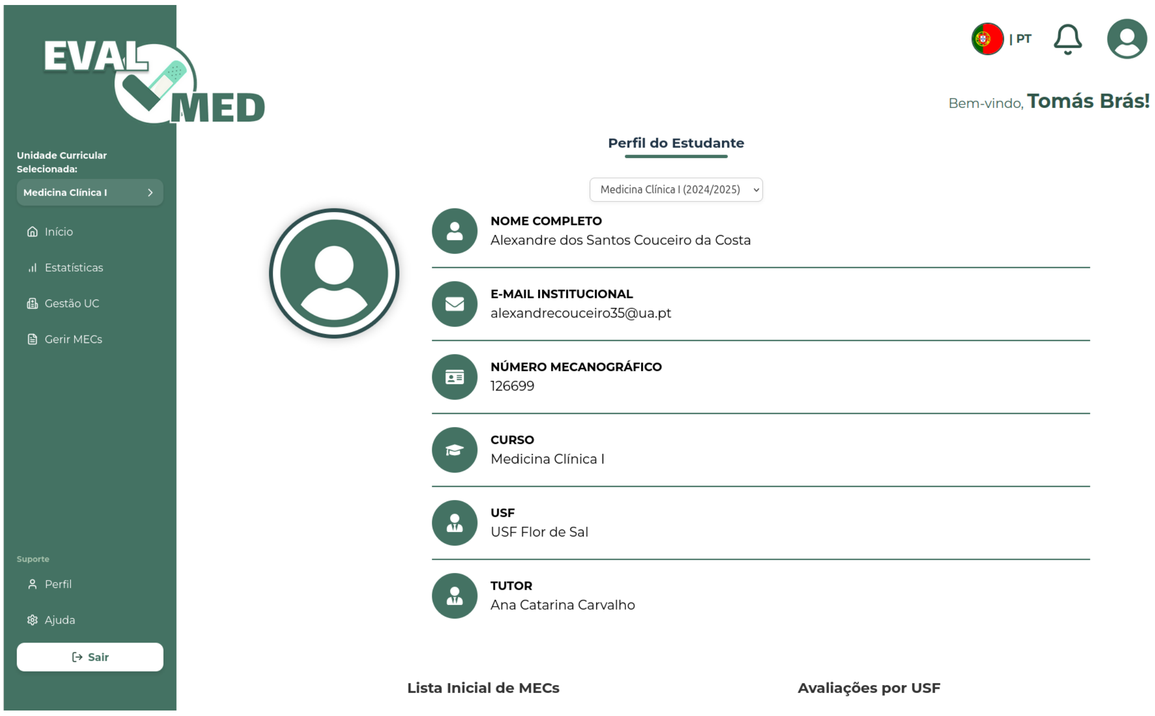
The layout of the administrator statistics page stands out for being highly organized and informative. On the left side, it displays the number of completed, pending, and improvement-required MECs, broken down by EPA-Nested and EPA, providing a detailed view of student progress.

On the right side, a learning curve chart visualizes the average scores obtained across different evaluations, allowing the administrator to assess whether student performance is improving or declining over time.

Lastly, a bar chart displays the number of feedback and reevaluation requests submitted by students, offering a comprehensive overview of engagement and interaction within the course.

- View individual student profiles

In order to have a more detailed view of each student's information and progress, the administrator can access the individual student profile.



NOME COMPLETO
Alexandre dos Santos Couceiro da Costa

E-MAIL INSTITUCIONAL
alexandrecouceiro35@ua.pt

NÚMERO MECANOGRÁFICO
126699

CURSO
Medicina Clínica I

USF
USF Flor de Sal

TUTOR
Ana Catarina Carvalho

Figure 4.22: Administrator – Student profile view

The profile includes the following personal and academic information:

- Full name
- Email address
- NMEC
- Curricular unit
- USF
- Assigned tutor

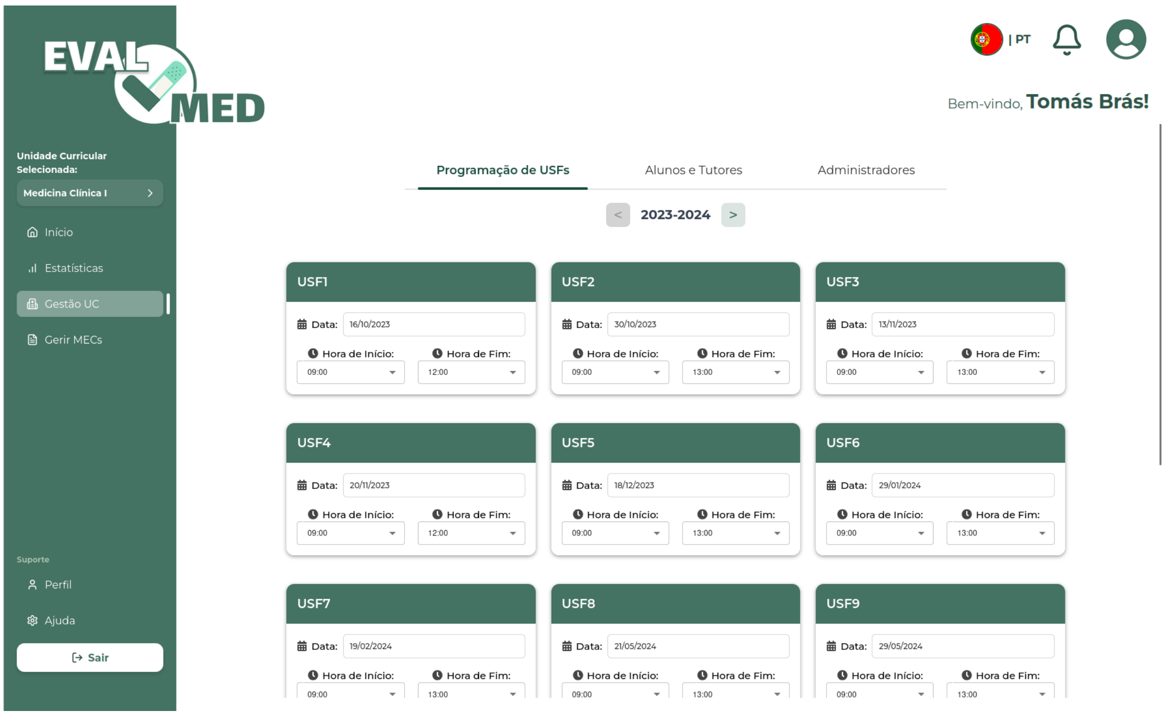
After viewing this data, the administrator can consult the student's academic progress, including:

- List of MECs where the student was approved, marked for improvement, or still pending
- Evaluation results obtained for each MEC
- Feedback submitted and received throughout the student's progression
- Reevaluation requests made by the student

The interface is designed to present all this information in a well-organized and clearly spaced layout. Given the volume of data related to each student, the design avoids overlapping elements and maintains a clean structure. The administrator sees a summary at the top, followed by vertically stacked cards for each data category. Each card includes a link or button to access more detailed views, allowing the administrator to navigate directly to the corresponding section for in-depth analysis.

- Add, remove, or edit evaluations

The administrator is responsible for setting up the academic years and, for each new year, generating a predefined number of evaluations. For each evaluation, the administrator must specify the location, date, and time.



USF1
Data: 16/10/2023
Hora de Início: 09:00
Hora de Fim: 12:00

USF2
Data: 30/10/2023
Hora de Início: 09:00
Hora de Fim: 13:00

USF3
Data: 13/11/2023
Hora de Início: 09:00
Hora de Fim: 13:00

USF4
Data: 20/11/2023
Hora de Início: 09:00
Hora de Fim: 12:00

USF5
Data: 18/12/2023
Hora de Início: 09:00
Hora de Fim: 13:00

USF6
Data: 29/01/2024
Hora de Início: 09:00
Hora de Fim: 13:00

USF7
Data: 19/02/2024
Hora de Início: 09:00
Hora de Fim: 13:00

USF8
Data: 21/05/2024
Hora de Início: 09:00
Hora de Fim: 13:00

USF9
Data: 29/05/2024
Hora de Início: 09:00
Hora de Fim: 13:00

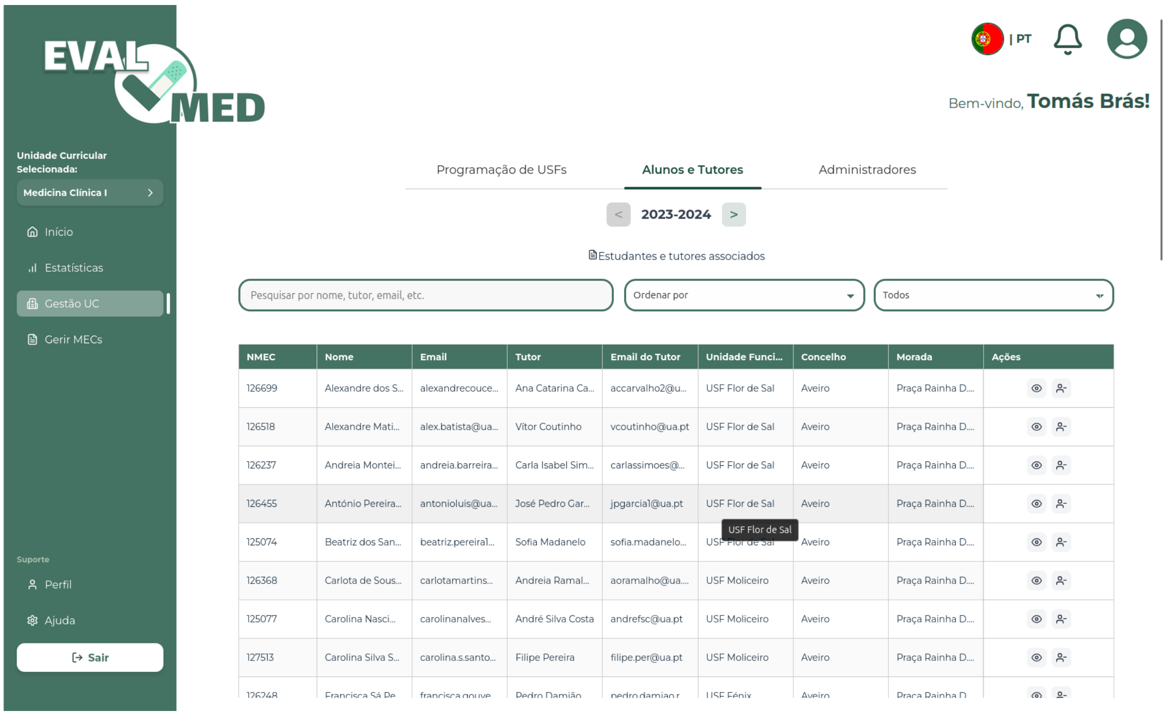
Figure 4.23: Administrator – Manage evaluations

Through the interface, the administrator can use the "+" button to add new evaluations. Each evaluation entry can be easily edited to update key information such as the date, time, and location. Once added, these evaluations are integrated into the system, making them available to tutors who can then proceed to plan each evaluation by assigning the appropriate MECs to their students.

- Manage students, tutors, and administrators

For each academic year, the administrator is responsible for managing the respective groups of students, tutors, and other administrators. Users can be added individually or in bulk through an Excel template. The administrator can download the pre-defined template, fill in the required columns, and then import it into the platform. During the import process, the system allows the administrator to map each column from the Excel file to the corresponding fields in the platform.

The data imported may include: student name, email, NMEC, tutor name, tutor email, municipality, address, and USF. Each user entry can be viewed in detail, edited, or deleted as needed.



NMEC	Nome	Email	Tutor	Email do Tutor	Unidade Funci...	Concelho	Morada	Ações
126699	Alexandre dos S...	alexandrecouce...	Ana Catarina Ca...	accarvalho2@u...	USF Flor de Sal	Aveiro	Praça Rainha D....	
126518	Alexandre Mati...	alex.batista@ua...	Vítor Coutinho	vcoutinho@ua.pt	USF Flor de Sal	Aveiro	Praça Rainha D....	
126237	Andreia Montei...	andreia.barreira...	Carla Isabel Sim...	carlassimoes@...	USF Flor de Sal	Aveiro	Praça Rainha D....	
126455	António Pereira...	antonioluis@ua...	José Pedro Gar...	jpgarcia1@ua.pt	USF Flor de Sal	Aveiro	Praça Rainha D....	
125074	Beatriz dos San...	beatriz.pereira1...	Sofia Madanelo	sofia.madanelo...	USF Flor de Sal	Aveiro	Praça Rainha D....	
126368	Carlota de Sous...	carlotamartins...	Andreia Ramal...	aoramalho@ua....	USF Moliceiro	Aveiro	Praça Rainha D....	
125077	Carolina Nasci...	carolinanalves...	André Silva Costa	andrefsc@ua.pt	USF Moliceiro	Aveiro	Praça Rainha D....	
127513	Carolina Silva S...	carolina.s.santo...	Filipe Pereira	filipe.per@ua.pt	USF Moliceiro	Aveiro	Praça Rainha D....	
126248	Francisca Sá Pe	francisca.gouve	Pedro Damião	pedro.damiao.r	USF Fénix	Aveiro	Praça Rainha D	

Figure 4.24: Administrator – User management interface

The interface is divided into three main tables:

- A table showing the associations between students and their respective tutors
- A table listing all students
- A table listing all tutors

Each table provides options for adding, editing, or removing entries. Buttons such as "Add Student" and "Add Tutor" open modals where the administrator can manually input user data directly into the platform.

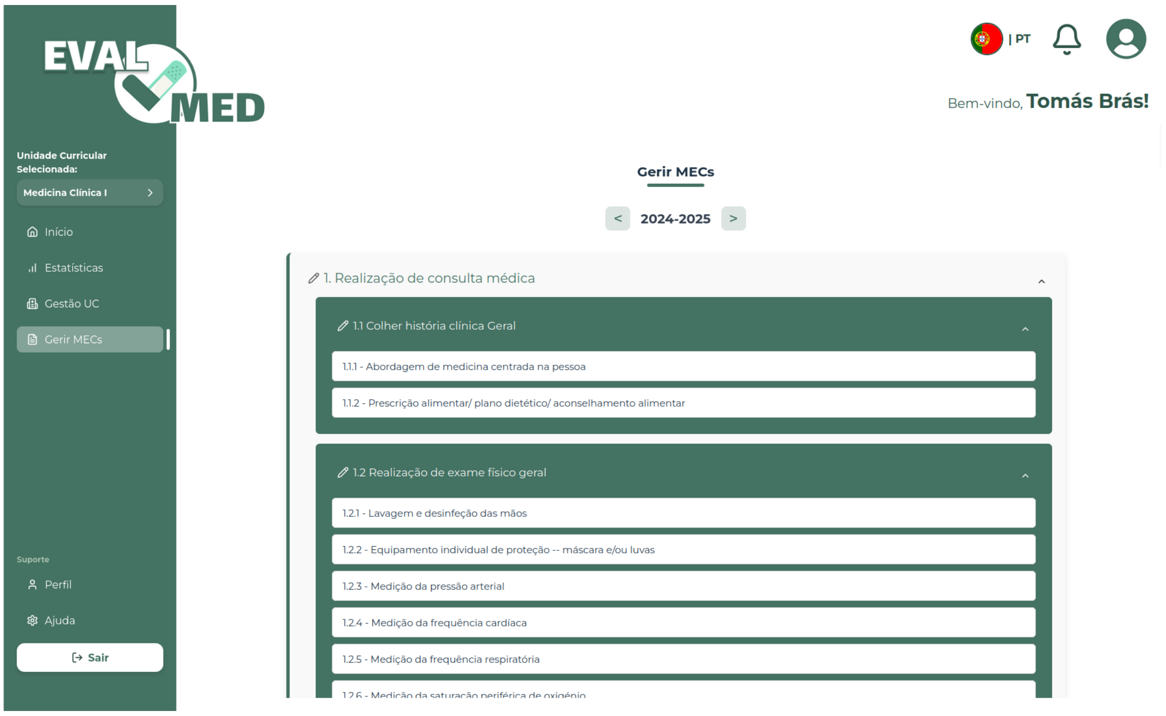
The system also allows for the creation of associations between students and tutors. Each student can only be assigned one tutor, and each tutor can only be assigned to one student per curricular unit. The platform enforces this restriction, ensuring that no tutor is assigned to multiple students within the same course.

- View and edit MEC procedures

The MECs that are part of the evaluations are added and managed by the administrator. When creating or editing a MEC, the administrator fills in key information such as:

- Version
- Title (Name)
- Steps involved
- Detailed procedures steps
- Instructional resources
- Bibliographic references

This setup allows tutors to select the appropriate MECs associated with the course and incorporate them into their evaluation plans.



1. Realização de consulta médica

1.1 Colher história clínica Geral

1.1.1 - Abordagem de medicina centrada na pessoa

1.1.2 - Prescrição alimentar/ plano dietético/ aconselhamento alimentar

1.2 Realização de exame físico geral

1.2.1 - Lavagem e desinfeção das mãos

1.2.2 - Equipamento individual de proteção -- máscara e/ou luvas

1.2.3 - Medição da pressão arterial

1.2.4 - Medição da frequência cardíaca

1.2.5 - Medição da frequência respiratória

1.2.6 - Medição da saturação periférica de oxigénio

Figure 4.25: Administrator – MEC procedures management

The interface is highly intuitive, featuring a pencil icon next to each EPA, EPA-Nested, and MEC. Administrators can click these icons to edit existing information or add new content, ensuring that the system remains up to date, accurate, and clearly organized for all users.

Table 4.1: Platform Functionalities by User Role

Students
• Self-assess upcoming evaluations • Request evaluation in future sessions • Request feedback from tutors • View performance statistics • Access educational materials • Check locations of upcoming evaluations • View past curricular year details • Send suggestions to the support team • See evaluation details from calendar • Receive notifications of important events • Download PDFs with protocol information
Tutors
• Evaluate students in real time • Provide feedback (on request or voluntarily) • Easily change student • Monitor student progress via statistics • Add protocols to a specific evaluation • Accept/Deny reevaluation requests • View past curricular year details • Send suggestions to the support team • Download PDFs with protocol information • Receive notifications of important events
Administrators
• Manage courses and associated EPAs • Import/Export course data via Excel • View overall course statistics • Create new evaluations for a year • Add a protocol by importing a PDF file • Add a new student/tutor manually • View past curricular year details • View student details • Manage course administrators

Other Functionalities for All Users

- Multi-language support
- Push notifications
- Offline synchronization of minimum features
- Voice-to-Text
- Support page

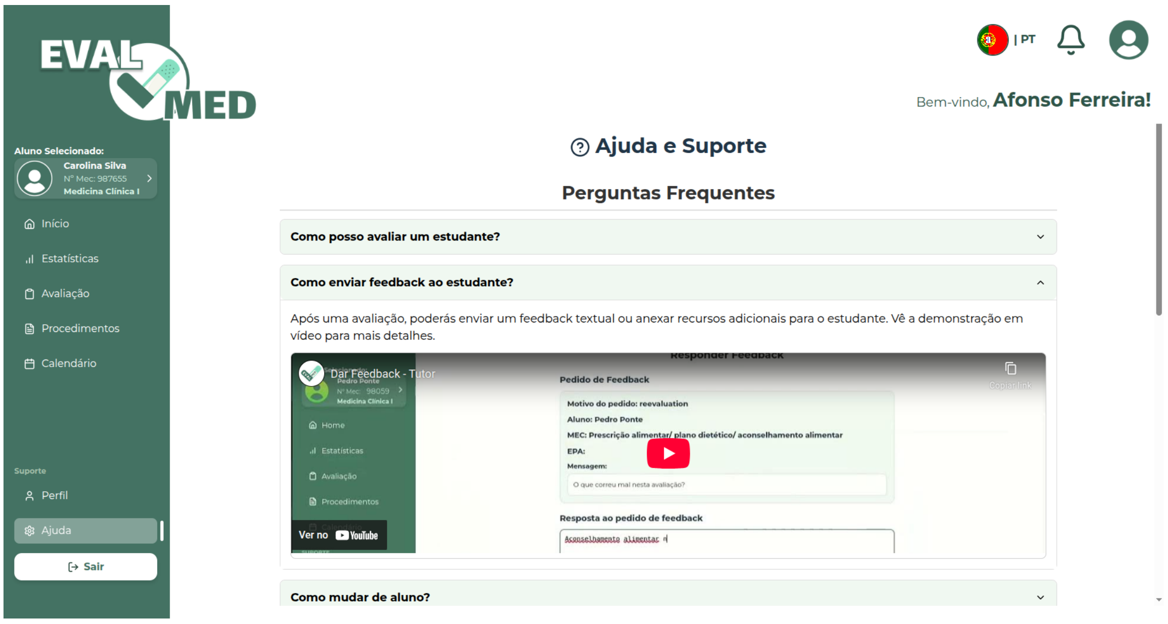
- Profile
- Send suggestions to the support team
- Notification of important events

4.1.3.10 Core Functionalities and Technical Implementation

- **Multi-language support**: Implemented using the library `react-i18next` (version 15.5.1). The translation files are stored in the `public/locales` directory as JSON files, organized by language. The application dynamically loads the appropriate translation file based on the user's selected language, enabling seamless multilingual support throughout the interface.
- **Voice-to-Text**: We integrated `react-speech-recognition`, a React library that provides a simple interface for leveraging the browser's native Speech Recognition API. This functionality is available not only to students but to any user interacting with input fields, such as tutors or administrators. It allows users to dictate their messages directly, enhancing accessibility and improving the user experience, especially in situations where typing is less convenient.
- **Icons**: All icons used across the application were implemented using `react-icons` (version 5.5.0). This library provides a unified way to access icon sets from multiple providers while keeping the codebase consistent and lightweight.

4.1.3.11 Support Page

We had initially planned to develop usability manuals and guides to assist users in navigating the application, since they are not familiar with this type of platform. However, when we were developing the application, we realized that some operations were very visual and had a long flow of steps, which made it difficult to explain in a manual. Therefore, we decided to create a help page that would provide users with the necessary information to use the application effectively. This help page is accessible from the side bar of the application and from the profile page of each user. It contains a list of tasks that the user can perform, along with a description of each task and a video uploaded in our YouTube channel [7] of the task being performed. This way, the user can easily find the information they need and see how to perform the task in practice. This page is different depending on the type of user, since the tasks that each user can perform are different.



Bem-vindo, **Afonso Ferreira!**

Ajuda e Suporte

Perguntas Frequentes

Como posso avaliar um estudante?

Como enviar feedback ao estudante?

Após uma avaliação, poderás enviar um feedback textual ou anexar recursos adicionais para o estudante. Vê a demonstração em vídeo para mais detalhes.

Responder Feedback

Pedido de Feedback

Motivo do pedido: reevaluation
Aluno: Pedro Ponte
MEC: Prescrição alimentar/ plano dietético/ aconselhamento alimentar
EPA:
Mensagem:
O que correu mal nesta avaliação?

Resposta ao pedido de feedback

Aconselhamento alimentar

Como mudar de aluno?

Figure 4.26: Example of the Tutor's support page

4.1.4 Backend

4.1.4.1 Folder Structure

Our structure for the backend includes our project folder and two configuration files:

```
└── backend/
    ├── src/main/java/pt/ua/epa/evalmed/
    |   ├── config/
    |   ├── controller/
    |   ├── dto/
    |   ├── model/
    |   |   └── mongo_documents/
    |   ├── repository/
    |   ├── scripts/
    |   |   └── pdfs/
    |   ├── service/
    |   |   └── impl/
    |   └── util/
    ├── Dockerfile.backend
    └── pom.xml
```

Figure 4.27: Folder structure of the application's backend

Starting with our condifuration files, *Dockerfile.backend* is used by our *docker-compose.yml* file to build the backend image. It uses the *openjdk:21-jdk* image as a base and the already built jar file to run the application.

Our *pom.xml* defines the dependencies and plugins used in our project. It includes the SpringBoot, email, database and other necessary dependencies.

The *src* folder contains the main code of our application, navigating throught to path *src/main/java/pt/ua/epa/evalmed* we can see the components of our application:

- **config**: Configuration files that define the application's behavior, including security settings, API configuration, and exception handling, includes *SecurityConfig.java* for authentication setup, *JwtAuthenticationFilter.java* for token validation, *MailConfig.java* for email service configuration, and *BackupScheduler.java* for automated database backups.
- **controller**: Our REST API endpoints that handle the requests made. The controllers are organized by domain with specific responsibilities, including *AuthController.java* for authentication, *EPAController.java* for entrustable professional activities management, *FeedbackController.java* for feedback operations, and role-specific controllers like *StudentController.java*, *TutorController.java*, and *AdministratorController.java.*
- **dto**: Contains Data Transfer Objects that facilitate the exchange of information between the client and server. These include specific purpose DTOs like *EvaluationDTO.java*, *FeedbackRequestDTO.java*, *MECCompletionStatisticsDTO.java*, and *StudentDetailsTableDTO.java* for structured data presentation. They were created to facilitate the transfer of data between the frontend and backend, ensuring that only necessary information is sent over the network, as some entitites contain large amounts of data that are dispensable for some operations.
- **model**: Defines the core domain entities of the application with their relationships. The model includes educational entities (*Course.java*, *EPA.java*, *MEC.java*), user types (*Student.java*, *Tutor.java*, *Administrator.java*), evaluation entities (*AutoEvaluation.java*, *TutorEvaluation.java*, *FinalEvaluation.java*), and supporting entities like *Feedback.java* and *Notification.java.*
  - **mongo_documents**: Contains the *MecDetails.java* document structure for MongoDB storage of detailed MEC information.
- **repository**: Contains interfaces that extend Spring Data's repository interfaces for database operations. Each entity has a corresponding repository, such as *StudentRepository.java*, *MECRepository.java*, *EvaluationRepository.java*, and *FeedbackRepository.java.* The repositories include custom query methods for specific data retrieval needs.
- **scripts**: Houses utility scripts for database population and maintenance.
  - **pdfs**: Contains MEC procedure PDFs that define the standards for each competency evaluation, such as *MEC_Exame objetivo do joelho.pdf* and *MEC_Medicina Centrada na Pessoa.pdf* (these files were provided to us by the Medicine course representatives). We then use *Python* scripts like *pdf_processer.py* to extract text from these PDFs and store it in the *MongoDB* database as documents.
- **service**: Defines interfaces for the application's business logic. Each service interface corresponds to a specific domain functionality, such as *StudentService.java*, *FeedbackService.java*, and *ReevaluationService.java.* These interfaces define methods for operations like student management, feedback submission, and reevaluation requests. The service layer abstracts the business logic from the controllers, promoting separation of concerns and making the codebase more maintainable.

  - **impl**: Contains concrete implementations of service interfaces. Notable specialized services include *OTTService.java* for one-time tokens and *SuggestionLimitService.java* for managing suggestion constraints.
- **util**: Contains the *DependencyAwareMysqlExporter.java* utility for database backup operations that respect table dependencies through topological sorting.

4.1.5 Nginx Configuration

Our Nginx setup functions as a reverse proxy that unifies access to all application components. The configuration implements secure HTTPS access with TLS 1.2/1.3, security headers, and dedicated routing paths for different services. It handles request forwarding to the React frontend, backend API endpoints, file uploads, and monitoring dashboards. The configuration also includes internal Docker network routing and proper header forwarding to maintain client context across all requests.

4.1.6 Backups

The backup system automates database preservation using the BackupScheduler component, which creates timestamped SQL dumps at scheduled intervals. The *DependencyAwareMysqlExporter* utility ensures proper handling of table relationships through topological sorting, ensuring a reliable restoration. Our built-in data retention policy automatically removes backups older than seven days to manage storage space efficiently. Backups are stored in a dedicated directory separate from application code to facilitate preservation during system updates.

4.1.7 API

All requests between the frontend and backend are performed through endpoints that follow REST API best practices. Every endpoint is documented using Swagger, and the complete API documentation is provided in Appendix A.

4.2 Notification System

4.2.1 Push Notifications

One of the requirements of the project was to implement a notification system that would allow users to receive different types of notifications. Therefore, we implemented a notification service that can be used in the business logic of the application to send notifications to specific users when certain events occur. These notifications are then displayed in the user interface in a dedicated page, but the user can also see the number of unread notifications in the header of the application, as seen in figure 4.28. If the user accepts the necessary permissions, they will also receive push notifications in their device when these events occur, even if the application is not open, due to FCM integration.

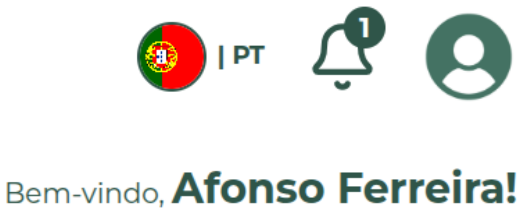


Figure 4.28: Notification header with the number of unread notifications

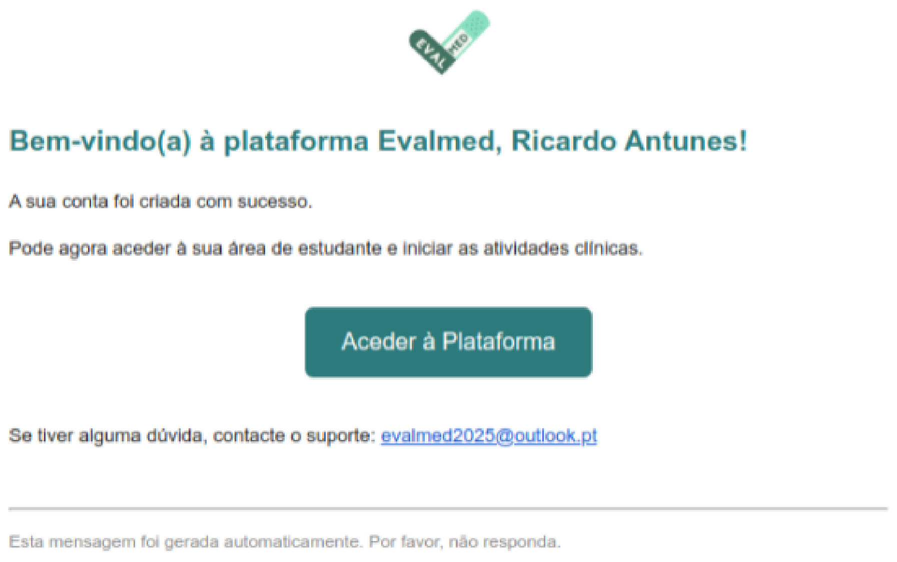
The events that trigger notifications are the following:

- **Administrator updates the time/date of an evaluation**: When the administrator changes the time or date of an evaluation, all students and tutors that are associated with that evaluation receive a notification informing them of the change.
- **Administrator assigns a tutor to a student**: When an administrator assigns a tutor to a student, both the student and the tutor receive a notification informing them of the assignment.
- **Tutor adds new MECs to an evaluation**: When a tutor adds new MECs to an evaluation, the student receives a notification informing them of the new MECs that will be part of the evaluation.
- **Tutor submits an evaluation**: When a tutor submits an evaluation for a student, the student receives a notification informing them that the evaluation has been submitted and is available for consultation.
- **Tutor submits a final evaluation**: When a tutor submits a final evaluation for a student, the student receives a notification informing them that the final evaluation has been submitted and is available for consultation.
- **Student submits a self-evaluation**: When a student submits a self-assessment for an evaluation, the tutor receives a notification informing them that the self-assessment has been submitted and is available for consultation.
- **Student asks for feedback**: When a student requests feedback from their tutor, the tutor receives a notification informing them of the request.

- **Tutor sends feedback**: When a tutor sends feedback to a student, the student receives a notification informing them of the feedback.
- **Student requests a reevaluation**: When a student requests a reevaluation, the tutor receives a notification informing them of the request.
- **Tutor accepts/denies a reevaluation request**: When a tutor accepts or denies a reevaluation request, the student receives a notification informing them of the decision.

4.2.2 Email Notifications

In addition to push notifications, we also implemented a simple email notifications system using JavaMailSender to send emails to users when they are first added to the platform. This is useful for example for when a administrator adds a new administrator, tutor or student to the platform, they are automatically sent an email informing that they can now access the platform. In image 4.29 we can see an example of an email notification sent to a user when they are added to the platform.


Bem-vindo(a) à plataforma Evalmed, Ricardo Antunes!

A sua conta foi criada com sucesso.

Pode agora aceder à sua área de estudante e iniciar as atividades clínicas.

Se tiver alguma dúvida, contacte o suporte: evalmed2025@outlook.pt

Esta mensagem foi gerada automaticamente. Por favor, não responda.

Figure 4.29: Email notification example

4.3 Authentication and Authorization

4.3.1 Implemented Authentication Methods

During the development of the project, three authentication methods were implemented to accommodate different environments and user needs. These methods are detailed below.

- **UA IdP Authentication**: This first method enables users to authenticate using their University of Aveiro credentials through the institution's IdP. It was implemented by following the guidelines provided in an internal GitHub repository **ua\_idp\_github** available to the University of Aveiro community. The integration was based on the OpenID Connect (OIDC) protocol, which extends OAuth 2.0 to include authentication capabilities. The implementation was successfully tested in a local development environment, allowing users to log in using their university credentials. However, due to challenges in obtaining official client credentials (client ID and secret) from Information and Communication Technologies services (sTIC), this method could not be used in the production environment. Nevertheless, the implementation remains in the codebase for potential future use.
- **One-Time Token Authentication**: This method allows users to log in using a one-time token sent to their email address. This token is generated by the backend and each token has a validity time of 3 minutes.
- **Elixir IdP Authentication**: This method allows authentication through the Elixir platform's Identity Provider. [8] It was the most recent authentication strategy added to the system and is actively used in production. It allows users to log in using other institutions' credentials, such as those from University of Aveiro or other official institutions that are part of the Elixir network, so it ends up being very similar to the first method. The main difference is that users need to be registered in the Elixir platform to be able to logi in. This registration can be done using institutional credentials or by creating a new account on the Elixir platform. The integration with the Elixir IdP was implemented using the OAuth 2.0 protocol. As previously mentioned, this method is currently in use in the production environment but it's expected to be removed if in the future the first method is successfully implemented with the official client credentials from sTIC.

4.3.2 Authentication Flows Overview

The authentication flows implemented for the second and third methods - OTT and Elixir IdP follow distinct mechanisms but ultimately serve the same purpose of securely identifying users and granting them access to the platform. Below is a detailed description of each flow, including the steps involved and the interactions between the frontend and backend components.

One-Time Token Authentication Flow:.

1. The user clicks the "Login with Email" button on the initial page.

2. The user enters their email address and clicks the "Submit" button.
3. If the email exists in the system, the backend generates a OTT and sends it to the user's email address and the user is redirected to the code input page.
4. The user enters the received token in the input field and clicks the "Confirm" button.
5. The frontend calls the backend endpoint `/api/auth/ott/token`, sending the token for validation.
6. If valid, the backend generates a session by issuing JSON Web Token (JWT) access and refresh tokens.
7. The frontend redirects the user to specific pages based on their role (Student, Tutor, or Administrator) and fetches user-specific information from the backend via the `/api/auth/user/me` endpoint.

Elixir Identity Provider Authentication Flow.

1. The user selects their institution from a dropdown menu on the initial page.
2. The user clicks on the "Continue" button, which redirects them to the Elixir IdP login page.
3. The user enters their credentials on the IdP page and, if successful, is redirected back to the callback URL with a code.
4. The frontend captures this code and sends it to the backend endpoint `/api/auth/token`, along with a code verifier (PKCE) for security.
5. The backend exchanges the code for an access token from the Elixir IdP, retrieves user information, in this case only the email, and checks if the user exists in the system.
6. If the user exists in the system, the backend issues local JWT access and refresh tokens and stores them as secure cookies.
7. The frontend redirects the user to their respective dashboard based on their role, fetching user-specific information from the backend via the `/api/auth/user/me` endpoint.

This flow can be different if it's the first time the user logs in with the Elixir Id

4.3.3 User Access Control

Although multiple authentication methods are supported, only users that have been previously registered in the platform's database are allowed to log in. This means that even if a user successfully authenticates via an external IdP, they will not be granted access to the platform unless the email used has been explicitly added to the platform's user database by an administrator. This is a deliberate design choice to ensure that users don't have access to resources or functionalities that they are not authorized to use.

4.3.4 Token Management and Session Handling

To ensure secure and scalable session management, the system uses a token-based authentication mechanism based on JWT's, applied to both the OTT flow and the OAuth2-based Elixir IdP integration.

Upon successful authentication, the backend generates two signed JWTs:

- An **access token**, valid for 1 hour, which includes the user's email and expiration time.

- A **refresh token**, valid for 14 days, used to request a new access token without requiring the user to log in again.

Both tokens are stored as HTTP-only cookies to prevent access via JavaScript protecting the system against XSS attacks. **clerk\_httponly\_2023** The access token is used for authenticating API requests, while the refresh token is used internally by the frontend to renew the access token when it expires.

To make the refresh process, the frontend uses an **Axios interceptor** to automatically handle token expiration. This interceptor captures failed HTTP responses, specifically those with status codes `401` (Unauthorized) and checks whether the original request has already been retried. If it hasn't, it attempts to refresh the access token by sending a request to the `/api/auth/refresh` endpoint using the refresh token stored in the HTTP-only cookie. If the refresh succeeds, the interceptor reissues the original request with the new access token. If no refresh token is found, or the refresh attempt fails, the user is redirected to the login page to login again.

To retrieve authenticated user information, the backend provides the `/api/auth/user/me` endpoint. This endpoint verifies the access token, retrieves the associated user from the database, and returns a structured payload based on their role (Student, Tutor, or Administrator). This is used by the frontend to load the appropriate user context and UI elements.

A dedicated `/api/auth/logout` endpoint is also provided to explicitly end the user session by clearing both the access and refresh token cookies.

4.4 Security Considerations

Ensuring the security of the EvalMed platform was a fundamental concern throughout the development process, given it deals with personal data. Multiple layers of protection were implemented to mitigate risks and safeguard user data, including both infrastructural and software-level measures.

4.4.1 Infrastructural Security

From an infrastructural standpoint, the application is hosted on a VM managed by the University of Aveiro at IEETA. Access to this VM is restricted to the internal UA network, meaning only devices connected to the university's network (either physically or using a Virtual Private Network (VPN)) are able to initiate connections. This network-level restriction acts as the first layer of defense, effectively reducing exposure to external threats.

Additionally, VM access is further protected by authentication credentials, ensuring that even users within the network must provide valid credentials to access the machine.

4.4.2 Software Security

4.4.2.1 Reverse Proxy and Rate Limiting

Nginx, our reverse proxy, plays an important role in improving the security of the platform by applying a number of protections at the network level. One of the key measures is rate limiting: we've configured it to allow up to 10 requests per second from each client, with an extra burst capacity of 20 requests. This helps protect against brute-force attacks and Denial of Service (DoS) attempts by limiting how much traffic any single client can generate.

We've also added a set of HTTP security headers to every response to guard against common web attacks. For example, the X-Content-Type-Options: `nosniff` header blocks the browser from trying to guess file types, which helps prevent certain types of code injection. The X-Frame-Options: `SAMEORIGIN` header stops other websites from embedding our pages in iframes, protecting against clickjacking. And X-XSS-Protection: `1; mode=block` tells the browser to automatically block detected XSS attacks.

Finally, we also don't allow requests to be bigger than 30MB, which helps prevent certain types of attacks that try to overwhelm the server with large payloads.

4.4.2.2 Authenticated API Requests

All the API Request are secured through token-based authentication. Each API request must include a valid **access token**, which is sent via HTTP-only cookies. Since none of our endpoints should be public except for the authentication endpoints, this ensures that only authenticated users can access the platform's resources. This token also helps the system identify the user making the request, which helped us implement RBAC which will be discussed in the next section.

4.4.2.3 RBAC

Since in our platform we deal with different user roles (Student, Tutor, and Administrator), one of our requiremnents in terms of security was to implement a RBAC system. This means that each user has a specific role that determines what actions they can perform and what resources they can access.

The RBAC system is implemented at the backend, where each API endpoint is protected by a Security Service that checks the user's role and specific details before allowing access. This security mechanism ensures that users can only access the data and perform the actions that are strictly relevant to their role within the system. For example, students can only view and interact with their own evaluations and progress, tutors have access to information related to the students they supervise, and administrators can only manage courses they are associated with.

To achieve this, the backend includes a centralized security component responsible for enforcing access policies. This component operates by retrieving the authenticated user from the request access cookie and evaluating whether the user meets the necessary conditions to access a specific resource. These conditions include confirming roles, ensuring that the user has the necessary permissions to perform the requested action and verifying relationships between two users, for example, checking if a tutor is associated with a student.

If a user attempts to perform an action they are not authorized to carry out, such as accessing data from another student or modifying course information, the system immediately rejects the request and returns an appropriate error response, typically a `403 Forbidden` status. The use of a centralized security service made the implementation of this RBAC system more efficient and maintainable, as it's easier to implement to new endpoints.

4.4.2.4 Frontend Security Measures

We also took care to implement security measures on the frontend to protect the application against common web vulnerabilities. Since the interface is built with React, we benefit from built-in protections like automatic escaping of user input to help prevent XSS attacks. On top of that, we added extra safeguards, for example, restricting access to administrator pages so that only authorized users can view or interact with them, keeping tutors and students out of pages that are not relevant to them. Besides that, when uploading profile pictures, the frontend only allows images with a maximum size of 5MB and only accepts files that are images, for example, JPEG or PNG files. This helps prevent users from uploading malicious files that could compromise the security of the application.

4.4.3 Data Security

As our solution proposes itself as a solution to replace paper-based methods to evaluate medical students, it's imperative that the data we store is secured and protected against potential malicious attacks.

4.4.3.1 Authentication

IdP Login

Despite not having the support we needed from sTIC, we mimmicked an authentication using the UA's IdP via LifeScience. This is not the best way to authenticate, as it may present a person-based security risks. It's known that people have the tendency to use the same password for the LifeScience login and their own institutional email, therefore an attacker can catch its pray using LifeScience and then login into our system using that person's credentials. Should this system actually go online, we would have support from sTIC, therefore removing this restriction from us.

OTT Login

As a initial workaround to the original authentication method planned, we developed an OTT Login system, where the token is sent via email to the user, which needs to insert the token in the frontend interface. This method still is vulnerable to a potential attack where the attacker has access to one's email, however. This token is only available for 3 minutes and only usable once. In future work, we can invest on making this method more robust, for example, preventing DDoS attacks from making this method inviable. Additionally, merging the IdP Login and the OTT Login into a single login (Multi-Factor Authentication (MFA)) could also improve security, as well as other methods such as different emails for the OTT login, all customizable by the user.

4.4.3.2 GDPR

This solution has a great potential to enter the market with great success, but since we are dealing with educational services, one of the biggest issues we face is following GDPR directives. Our current database does follow these regulations, as the amount of effort and time we would need to strictly follow those would overshadow the entire project, which isn't the intended goal. However, we should we commercialize the product, we can make a roadmap of changes we need to make to our system in order to follow those regulations.

Covered data

The GDPR is meant to protect personal data for each individual, which in our case involves names, NMEC's, emails, and anything that can be used to **identify** someone. As such, the changes we need to perform should be oriented towards **pseudo-anonimyzing** all this data.

Changes to Database (DB) scheme

Each Person is currently directly tied down to its personal information, so we would need to create a different database where this information is encrypted, and then allow it to be referenced by our original database which will only process this data. This strategy is also used by **Moodle**, which uses this structure not only to oblige by GDPR directives, but also to offer 2 different business models to every client, something we could take for ourselves as well.

Responses to Legal Requests

Everyone has the right to delete their data from our platform - and everyone must also consent to our storage and use of their personal information. As such, not only do we need to implement mechanisms to safely delete someone's information from the system, but also only store this personal information if the person gives their consent.

In terms of creating a new account, which can't be done autonomously, and instead is done by an administrator, this account should only be active whenever the user accepts such. For this, we already have a welcome email mechanism implemented, so it would only be a matter of only storing the data (or storing it temporarily until the user confirms the consent) once the user consents to it (using the application and creating an account can count as a means of consent, if it is explicit).

When a user requests to delete their personal information from the system, they must be informed that this impedes their access to the service, and all evaluation history will be lost. If the user consents to this, then the information in the encrypted DB should all be deleted, keeping only the reference to it for history-analysis purposes, and whenever a different user comes across this account in the application it should merely say "Deleted Account" instead of the name of the person. This way we follow GDPR directives while also still mantaining history of events in the system, as we promise in many of our requirements.

4.5 Observability and Monitoring

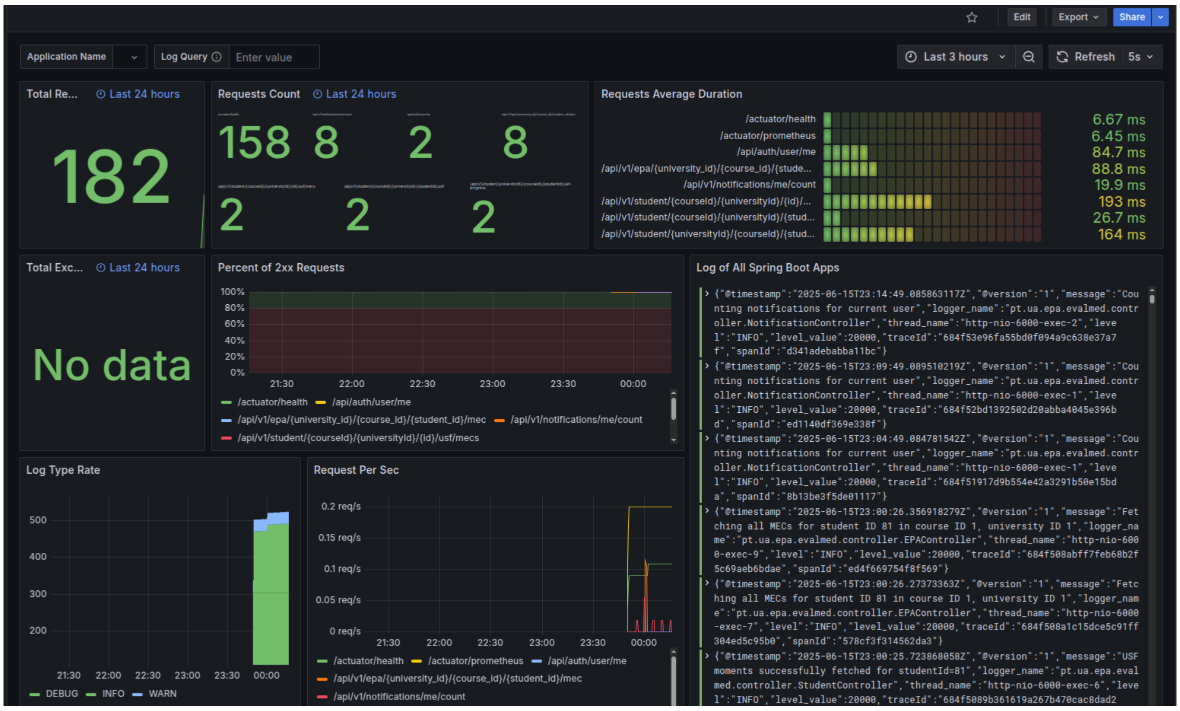
To ensure the stability, performance, and health of the EvalMed platform, we implemented a complete observability and monitoring solution using a combination of Prometheus, Grafana, Loki, and Promtail. This stack allows us to collect, visualize, and analyze metrics and logs from the backend containers, providing real-time insights into the system's behavior and performance.

Prometheus is responsible for collecting metrics from the backend application. These metrics are exposed via the Spring Boot Actuator at the `/actuator/prometheus` endpoint. It fetches metrics at a regular interval of 5 seconds, ensuring that we have up-to-date information about the application's performance.

Loki complements Prometheus by enabling centralized log aggregation and querying. Instead of collecting logs in the traditional way, Loki is designed to work efficiently with logs from containerized environments like Docker. Logs from the EvalMed backend are written to a shared volume and then ingested by Loki, allowing us to query them in Grafana alongside the metrics collected by Prometheus. This allows us to see for example the most recent logs from the backend application, the number of logs per level (e.g., INFO, ERROR), and the total number of logs generated over time.

Promtail acts as the agent that reads the log files generated by the backend application and forwards them to Loki. It is configured to watch the directory where the backend writes its logs (e.g., `uploads/backend.log`) and attaches relevant labels such as the job name in this case `backend`.

Grafana is used to visualize these metrics in a dashboard. We used a Grafana dashboard retrieved from the official Grafana repository [9] that was made to be used with Spring Boot applications and uses as Data Source both Prometheus and Loki. In figure 4.30, there's a


Edit
Export
Share
Application Name
Log Query
Enter value
Last 3 hours
Refresh 5s
Total Re... Last 24 hours
182
Requests Count Last 24 hours
158 8 2 8
2 2 2
Requests Average Duration
/actuator/health 6.67 ms
/actuator/prometheus 6.45 ms
/api/auth/user/me 84.7 ms
/api/v1/epa/{university_id}/{course_id}/{stude... 88.8 ms
/api/v1/notifications/me/count 19.9 ms
/api/v1/student/{courseId}/{universityId}/{id}/... 193 ms
/api/v1/student/{courseId}/{universityId}/{stud... 26.7 ms
/api/v1/student/{universityId}/{courseId}/{stud... 164 ms
Total Exc... Last 24 hours
No data
Percent of 2xx Requests
/actuator/health
/api/auth/user/me
/api/v1/epa/{university_id}/{course_id}/{student_id}/mec
/api/v1/notifications/me/count
/api/v1/student/{courseId}/{universityId}/{id}/usf/mecs
Log Type Rate
DEBUG
INFO
WARN
Request Per Sec
/actuator/health
/actuator/prometheus
/api/auth/user/me
/api/v1/epa/{university_id}/{course_id}/{student_id}/mec
/api/v1/notifications/me/count
Log of All Spring Boot Apps

```
> {"@timestamp":"2025-06-15T23:14:49.085863117Z","@version":"1","message":"Cou
  nting notifications for current user","logger_name":"pt.ua.epa.evalmed.contr
  oller.NotificationController","thread_name":"http-nio-6000-exec-2","leve
  l":"INFO","level_value":20000,"traceId":"684f53e96fa55bd0f094a9c638e37a7
  f","spanId":"d341adebabba11bc"}
> {"@timestamp":"2025-06-15T23:09:49.089510219Z","@version":"1","message":"Cou
  nting notifications for current user","logger_name":"pt.ua.epa.evalmed.contr
  oller.NotificationController","thread_name":"http-nio-6000-exec-1","leve
  l":"INFO","level_value":20000,"traceId":"684f52bd1392502d20abba4045e396b
  d","spanId":"ed1140df369e338f"}
> {"@timestamp":"2025-06-15T23:04:49.084781542Z","@version":"1","message":"Cou
  nting notifications for current user","logger_name":"pt.ua.epa.evalmed.contr
  oller.NotificationController","thread_name":"http-nio-6000-exec-1","leve
  l":"INFO","level_value":20000,"traceId":"684f51917d9b554e42a3291b50e15bd
  a","spanId":"8b13be3f5de01117"}
> {"@timestamp":"2025-06-15T23:00:26.356918279Z","@version":"1","message":"Fet
  ching all MECs for student ID 81 in course ID 1, university ID 1","logger_na
  me":"pt.ua.epa.evalmed.controller.EPAController","thread_name":"http-nio-600
  0-exec-9","level":"INFO","level_value":20000,"traceId":"684f508abff7feb68b2f
  5c69aeb6bdae","spanId":"ed4f669754f8f569"}
> {"@timestamp":"2025-06-15T23:00:26.27373363Z","@version":"1","message":"Fetc
  hing all MECs for student ID 81 in course ID 1, university ID 1","logger_nam
  e":"pt.ua.epa.evalmed.controller.EPAController","thread_name":"http-nio-6000
  -exec-7","level":"INFO","level_value":20000,"traceId":"684f508a1c15dce5c91ff
  304ed5c95b0","spanId":"578cf3f314562da3"}
> {"@timestamp":"2025-06-15T23:00:25.723868058Z","@version":"1","message":"USF
  moments successfully fetched for studentId=81","logger_name":"pt.ua.epa.eval
  med.controller.StudentController","thread_name":"http-nio-6000-exec-6","leve
  l":"INFO","level_value":20000,"traceId":"684f5089b361619a267b470cac8dad2
```

Figure 4.30: Grafana Dashboard for Monitoring Backend Services

screenshot of the Grafana dashboard with the metrics and logs collected from the backend application which can be accessed at the URL `http://evalmed.pt/grafana`.

In the future we plan to extend this observability solution to include the frontend application as well, to study the most accessed pages, the number of users online, most common errors, and other relevant metrics that can help us improve the user experience and performance of the platform.

4.6 CI/CD Workflows

During the development of this project, we implemented two CI workflows and a CD workflow using Github Actions to guarantee the reliability of our code and the quality of our application. Below is a description of the workflows created:

- **Build and Test Workflow**: The verification workflow runs on pushes and pull requests to master and develop. It ensures the application builds successfully, loads the correct configuration, and starts all services. It also verifies service health and shuts everything down cleanly afterward.
- **Java Formatter Workflow**: This workflow runs on pushes and pull requests to all branches. It enforces consistent code style across the backend by applying the Google Java Format tool. If changes are needed, it commits and pushes them automatically.
- **Deploy Workflow**: The deployment workflow runs on pushes to the master branch. It automatically builds the application, generates environment and configuration files using GitHub secrets, and redeploys the updated version in our IEETA VM.

We tried to implement a workflow that would run a SonarQube analysis on every push to the master branch, but we encountered some issues with the SonarQube server due to the repository being in a private organization. As a result, we were unable to implement this workflow.

CHAPTER 5

Tests and Results

5.1 Introduction

To test the usability and performance of EvalMed, different testing phases were carried out during the development of the application, which included both usability and load testing. Usability testing was conducted in two stages: first with a low-fidelity prototype and later with a high-fidelity version of the system. These sessions were crucial for identifying usability issues, navigation challenges, and conceptual misunderstandings. The feedback gathered allowed the team to improve the interface, adjust features to meet user expectations, and enhance the overall user experience. Validating design decisions with real users at different stages helped ensure a more user-centered and efficient product, while reducing the likelihood of costly revisions later on.

All usability tests were conducted in person at the Escola Superior de Saúde da Universidade de Aveiro (ESSUA). The low-fidelity tests took place early in the semester, on March 5th and 7th, 2025, while the high-fidelity tests, using the fully implemented application, were held on May 7th.

Load testing complemented this process by evaluating system performance under stress. Automated tests using the K6 tool successfully simulated multiple concurrent users and provided detailed performance metrics. Additionally, a real-world session was held in which a large group of students used the application simultaneously. This live scenario offered valuable insight into system stability under actual conditions, with the application remaining stable and responsive throughout. Further details are provided in the dedicated load testing section.

5.2 Usability Testing

Once the application requirements were defined, usability testing was carried out in two phases: first with a low-fidelity prototype, and later with a high-fidelity version of the system. The low-fidelity prototype served as an early representation of the interface, allowing for quick and cost-effective evaluation of the overall design before any code was written. This approach helped identify potential usability issues, gather user impressions, and validate the initial structure of the interface. Following iterative improvements and the implementation of core functionalities, a high-fidelity prototype was tested. This version reflected the final look and behavior of the application, enabling a more accurate assessment of the system's usability and performance in near-real conditions. Both phases of testing were essential to ensure a user-centered design, guiding the development process with continuous feedback and validation.

Throughout both usability testing phases, we held several meetings with the Medicine course coordinators, Juliana Sá and Patrícia Pinho, who provided continuous guidance and constructive feedback regarding the features and workflows of the platform. Their contributions were critical in ensuring the system remained aligned with the academic and clinical context for which it was being designed. The coordination and recruitment of participants for all test phases were supported by Professor Sílvia Badarra, whose involvement was fundamental to facilitating access to both students and tutors. These collaborative efforts helped validate the platform's goals and informed key design decisions.

5.2.1 Low-Fidelity Prototype Testing

5.2.1.1 Sample

The low-fidelity usability tests were conducted on March 5th and 7th, 2025, at the ESSUA. The participant sample consisted of 11 individuals: 6 students enrolled in the Clinical Medicine I course and 5 Medicine Tutors. These participants represented key target users of the platform, allowing us to evaluate the interface from both the student and teaching perspectives.

The recruitment of participants was made possible through the direct support of Professor Sílvia Badarra and in coordination with the Medicine course leadership. All participants were informed of the nature and purpose of the study and signed consent forms before starting the session (see Appendix I and J for the consent form). The consent included the use of personal data as well as image rights for potential use in future promotional or dissemination activities. The tests were conducted in a controlled environment, ensuring that participants could focus on the tasks without external distractions.

It is expected that some of the students involved in these tests may become future users of the system. This made their feedback particularly valuable, as their ability to successfully navigate and understand the platform indicates that the intended target audience is also likely to use the system effectively.

5.2.1.2 Method

The low-fidelity prototype was designed using the web-based design tool Figma [10]. The full prototype can be accessed through the following link. Prior to beginning the testing session, each participant was given a brief contextual explanation by the project team. This introduction included a presentation of the system's purpose, the goals of the testing phase, and an overview of the tasks to be performed.

Participants were informed that any difficulties they encountered during the process would be considered issues related to the interface itself and not the result of user error. This clarification was intended to ensure participants felt comfortable and confident throughout the session, creating a relaxed testing environment that encouraged honest interaction with the prototype.

Each participant was asked to complete five predefined tasks, detailed in both Appendix B and Appendix C, while being observed by a member of the testing team. During the tasks, observers remained close by and followed a standardized Observation Guide, provided in Appendix D, to ensure consistent data collection across sessions. Observers took real-time notes on user behavior, paying close attention to moments of hesitation, confusion, or error. Importantly, no interpretations or assumptions were made about participant behavior; only explicit verbalizations and actions were recorded.

At the conclusion of the task execution, participants were asked to complete a post-task questionnaire, available in Appendix E. This questionnaire allowed users to reflect on their experience, offer written comments, and respond to multiple-choice items. The final section of the questionnaire included the ten questions of the SUS[11], a widely recognized instrument for measuring perceived usability. This allowed the team to obtain both qualitative insights and a quantitative metric to evaluate the prototype's usability.

5.2.1.3 Results

After conducting the usability tests, we proceeded to analyze the data collected from the sessions. This analysis included not only the direct responses to the tasks but also observer notes and spontaneous comments made by the participants. These insights were essential for identifying usability issues, assessing how intuitively users interacted with the prototype, and gathering suggestions for improvement.

Student Interface. To test the student interface, five key tasks were defined, each designed to evaluate common actions expected from students within the platform. These tasks aimed to assess the clarity of navigation, feedback mechanisms, and content accessibility. The tasks and their respective goals were as follows:

- **Task 1:** Perform a self-assessment for the upcoming USF evaluation using the following autonomy levels:
  - MEC14 – N1
  - MEC15 – N3
  - MEC16 – N3

  Participants were asked to note the system's response after submitting the self-assessment.
- **Task 2:** Check the evaluation obtained in USF4, specifically identifying the autonomy level achieved in MEC3.
- **Task 3:** View the number of approved MEC points and identify the total approved value.

- **Task 4:** Access the procedure for MEC1 ("Person-Centered Medical Approach") and determine how many steps are included.
- **Task 5:** Request a re-evaluation of MEC3 and observe the feedback provided by the system.

Task Difficulty Assessment – Student Interface

The following chart presents the perceived difficulty of each student task. The values represent the best, average and worst ratings given by participants for each task.

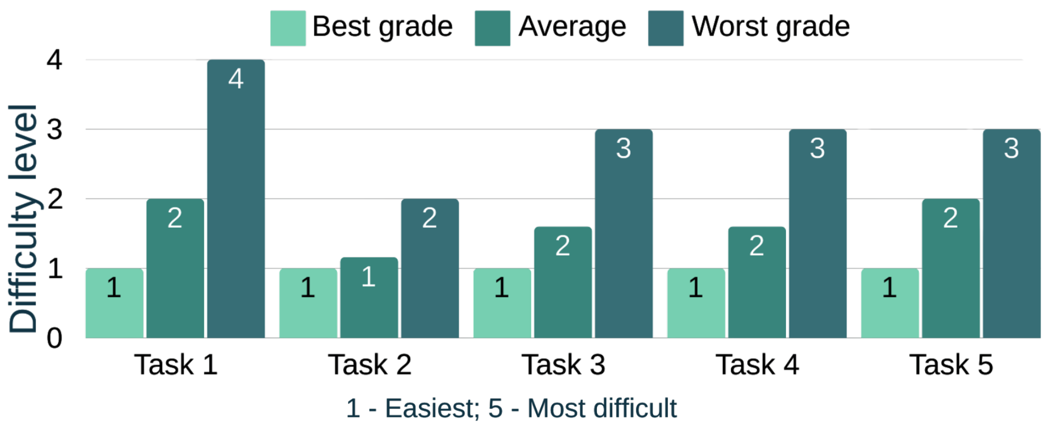


Figure 5.1: Perceived difficulty ratings for student tasks (best, average, worst).

Tutor Interface. Similarly, the tutor prototype was tested through a set of five tasks representing typical tutor responsibilities within the system. The tasks and their goals were as follows:

- **Task 1:** Check when the next student evaluation is scheduled, identifying the date and time.
- **Task 2:** Perform an evaluation for USF5 for the student "Maria do Mar" using the following autonomy levels:
  - MEC14 – N1
  - MEC15 – N2
  - MEC16 – N2

  Participants were then asked to describe the feedback provided after submitting the evaluation.
- **Task 3:** View the procedure for MEC1 and determine how many steps are presented.
- **Task 4:** Access the calendar and identify the date of the next scheduled evaluation.

- **Task 5:** Approve a student's request to re-evaluate MECs in the next USF and observe the feedback generated by the system.

Task Difficulty Assessment – Tutor Interface

The following chart shows the perceived difficulty of each tutor task, based on the best, average and worst scores assigned by the participants.

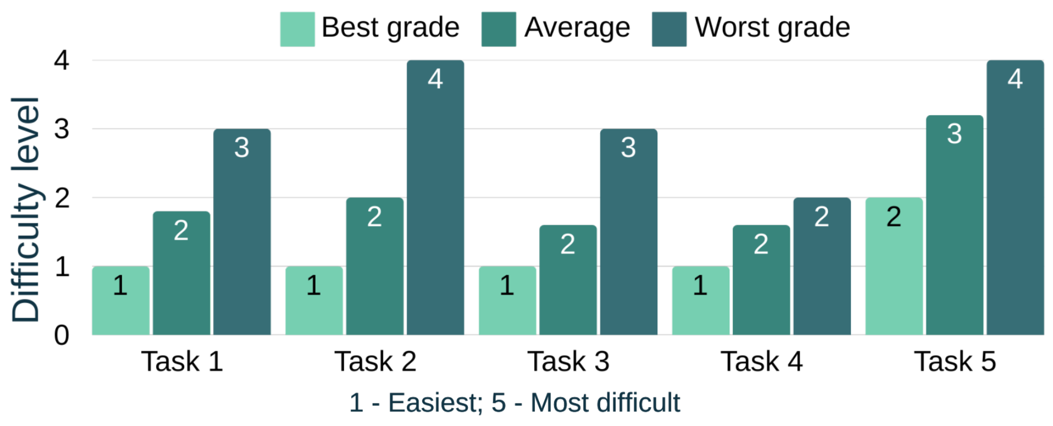


Figure 5.2: Perceived difficulty ratings for tutor tasks (best, average, worst).

During each usability test session, a member of the project team filled out an observer form while the participant performed the assigned tasks. This form included both quantitative and qualitative metrics, such as whether the task was completed, the time taken, the number of errors observed, signs of confusion, whether help was requested, and a difficulty score based on the observer's perception.

This data provides valuable insights into how intuitive and accessible the interface was for the users, and complements the self-reported post-task feedback. The analysis below reflects data gathered for each of the five tasks for overall usability.

Task Success

In those tasks, participants were evaluated based on whether they achieved the correct results. As illustrated in Figure 5.3, the overall rate of task success, defined as reaching the correct conclusion, was 92.7%.

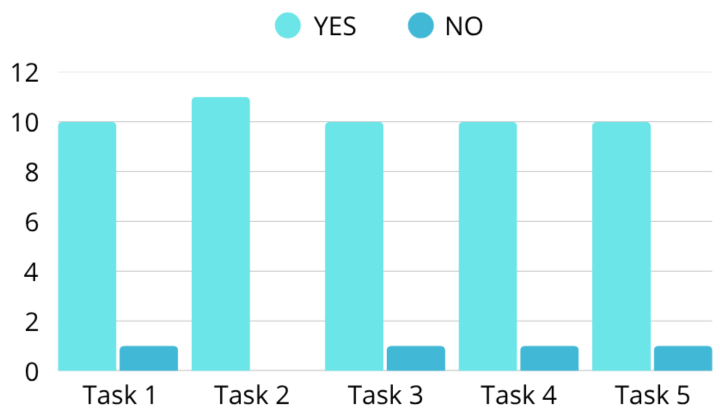


Figure 5.3: Tasks completed with correct results during usability tests.

Observer Data Tables

The following tables summarize the observational data for each task, categorized by participant role. Each table includes:

- Task completion status
- Time taken
- Number of errors
- Degree of confusion
- Whether help was requested
- Difficulty level (rated by observer)
- Additional notes

Table 5.1: Average time (min:sec) and errors per task (n = 5)

Task	Average Time	Average Error
Task 1	1:29	0.6
Task 2	1:13	0.4
Task 3	0:58	0.4
Task 4	0:49	0.2
Task 5	1:11	1.0

Table 5.2: Average perceived difficulty per task (1 = Very easy, 5 = Very difficult)

Task	Average Difficulty
Task 1	2.0
Task 2	1.6
Task 3	2.4
Task 4	2.2
Task 5	2.6

SUS

As mentioned in the previous subsection, participants completed a post-task questionnaire which included the SUS[11]. Figure 5.4 presents the median score for each question. The median was chosen over the average to minimize distortion from potential input errors.

Note that in SUS scoring, odd-numbered questions are positively worded, and even-numbered ones are negative. As such, the ideal scores are inverted: 5 for even-numbered questions and 1 for odd-numbered ones. This method reduces the risk of participants giving identical scores down the column out of habit.

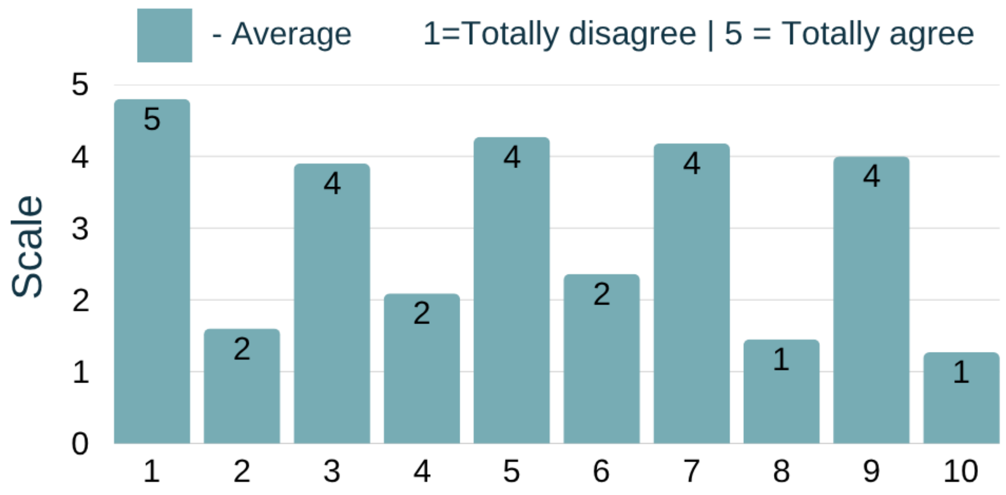


Figure 5.4: Median score per question on the SUS.

With these data, we calculated our SUS, obtaining a score of 81. Since this value is above the standard benchmark of 68, we can consider the system to be usable.

Regarding user comments and annotations about the system, we extracted the following summarized considerations:

- **Tutor Prototype**
  - **Task completion color differentiation and text inconsistency:** The homepage should clearly differentiate upcoming and past evaluations using distinct colors and consistent text sizes.
  - **Difficulty finding the re-evaluation request section:** The "Requests" section should clearly specify request and validation dates to avoid confusion.
  - **Calendar usability:** Users had trouble locating and understanding evaluation identifiers in the calendar view.
  - **MEC search button:** The search button in the navigation bar is not intuitive, leading to user confusion.
  - **Strengths:**
    - Tutors appreciated the statistics feature, finding it informative and engaging.
    - The evaluation forms were comprehensive and visually accessible, with effective use of colors.
- **Student Prototype**

  - **Highlight key points in evaluations:** Students often struggled when filling in strengths and weaknesses; autocomplete suggestions would help guide the input.
  - **Access to re-evaluation requests:** The re-evaluation request button on the homepage was not intuitive; additional access points should be available within the evaluation section.
  - **Strengths:**
    * The statistics feature was also highly valued by students for its clarity and usefulness.

Overall, users found the prototype intuitive, easy to use, and well-structured. Although some felt slightly overwhelmed at the beginning, most adapted quickly.

5.2.2 High-Fidelity Prototype Testing

5.2.2.1 Sample

In this new phase of testing, the aim was to evaluate the finalized application across the three main user roles implemented in our solution: Student, Tutor, and Administrator. Unfortunately, due to unforeseen communication challenges and technical issues related to email delivery, the recruitment process did not proceed as initially planned. Specifically, Professor Sílvia Badarra, who was responsible for coordinating participant recruitment among students and faculty for the Tutor and Administrator roles, was only able to secure the participation of four students at short notice.

Before the testing session starts, all participants were informed of the nature and purpose of the study and signed consent forms before starting the session (see Appendix I and J for the consent form). The consent included the use of personal data as well as image rights for potential use in future promotional or dissemination activities. The tests were conducted in a controlled environment, ensuring that participants could focus on the tasks without external distractions.

Although the original intention was to conduct usability testing with a broader and more diverse sample across all roles, the final participant pool was limited. Despite the smaller sample size, the feedback gathered proved valuable and provided meaningful insights into the usability and functionality of the system. While this sample cannot be considered fully representative, the observations and comments collected offer important perspectives to guide further development and refinement.

5.2.2.2 Method

The high-fidelity prototype consisted of a fully developed application encompassing the finalized functionalities for the Student, Tutor, and Administrator roles. Prior to each testing session, participants received a concise introduction by the project team, outlining the system's objectives, the specific goals of this testing phase, and a summary of the tasks to be undertaken.

Given the limited sample size, the testing sessions were adapted to focus primarily on the Student role, for which four participants were available. Each participant was asked to perform a series of tasks F designed to evaluate key interactions and features within the application. Similar to the low-fidelity phase, participants were reassured that any difficulties encountered would be attributed to the interface design rather than user error, promoting a relaxed and open testing environment.

Observers from the project team monitored the sessions closely, following a standardized Observation Guide (see Appendix G) to ensure consistent and objective data collection. Real-time notes were taken, focusing on moments of hesitation, confusion, errors, and requests for assistance. As before, only direct participant actions and verbalizations were recorded, avoiding subjective interpretation.

At the end of the tasks, participants completed a post-task questionnaire (see Appendix E) designed to gather qualitative feedback and usability perceptions. This questionnaire also included the SUS[11] to enable quantitative evaluation of the prototype's usability. Despite the smaller number of participants, the collected data provided valuable insights into the application's effectiveness and areas for improvement.

5.2.2.3 Results

After conducting the high-fidelity usability tests, we analyzed the data collected from participant responses, observer notes, and spontaneous comments. Although the number of participants was limited, the results provided valuable insights into the usability of the final application for the Student role.

Student Interface. The evaluation focused on five key tasks representing typical student interactions within the system. These tasks aimed to assess navigation clarity, feedback accuracy, and overall user experience. The tasks and objectives were:

- **Task 1:** Identify the date of the next scheduled evaluation.
- **Task 2:** Complete a self-assessment for the upcoming evaluation using autonomy levels selected by the participant.
- **Task 3:** Review the evaluation results for a specific evaluation (e.g., USF4) and identify the autonomy level achieved in the first MEC evaluated.
- **Task 4:** Locate and view the procedure and explanatory video for a specified MEC.
- **Task 5:** View the procedure and explanatory video for the Marco Educativo Clínico (MEC) 1.2.3.
- **Task 6:** Request a re-evaluation for a MEC and observe the system's feedback.

Task Difficulty and Feedback. Participants rated the difficulty of each task from 1 (Very Easy) to 5 (Very Difficult) and provided qualitative comments. Key observations from the data include:

- **Task 1 (Next Evaluation Date):** All participants easily found the date (difficulty rating 1). The date was typically located in the "Avaliações" page or on the Home calendar.
- **Task 2 (Self-Assessment Submission):** Participants encountered some difficulties due to the form being inaccessible before a certain date or after a time limit. Difficulty

ratings ranged from 1 to 4. Some users mentioned unclear feedback messages such as "Ainda não é possível submeter a autoavaliação" or "Formulário encerrado".

- **Task 3 (Evaluation Results):** Most participants could identify the autonomy level for the first MEC evaluated, though some had moderate difficulty (ratings 1 to 3). One participant reported difficulty finding the MEC due to lack of clarity on filtering mechanisms.
- **Task 4 (Procedure and Video):** Participants generally found the MEC procedure video easily (difficulty mostly 1), but some noted UI issues such as buttons or navigation elements being obscured by welcome banners or interface elements.
- **Task 5 (Procedure and Video for MEC 1.2.3):** Participants consistently found the procedure and video easily (difficulty mostly 1). The video titled "Como medir a tensão arterial" was correctly identified by all participants. No major usability issues were reported for this task.
- **Task 6 (Re-evaluation Request):** Most participants successfully submitted re-evaluations, receiving confirmation messages such as "Reavaliações submetidas com sucesso". However, some experienced interface issues where the submit button was partially hidden by navigation bars, causing momentary confusion and increased difficulty ratings (1 to 3). One participant reported an error when submitting.

Qualitative Feedback Highlights.

- The submit button was sometimes hidden behind interface elements (welcome banners or bottom bars), complicating task completion.
- Filtering mechanisms for MEC items were not immediately obvious, causing some users to struggle with Task 3.
- Feedback messages post-submission were generally clear but could benefit from better visibility and clearer wording to reduce user uncertainty.
- The video tutorials were highly appreciated as useful aids to understand clinical procedures.

Despite these minor usability issues, participants generally found the system intuitive and the tasks manageable. This feedback offers valuable guidance for further refinements of the interface.

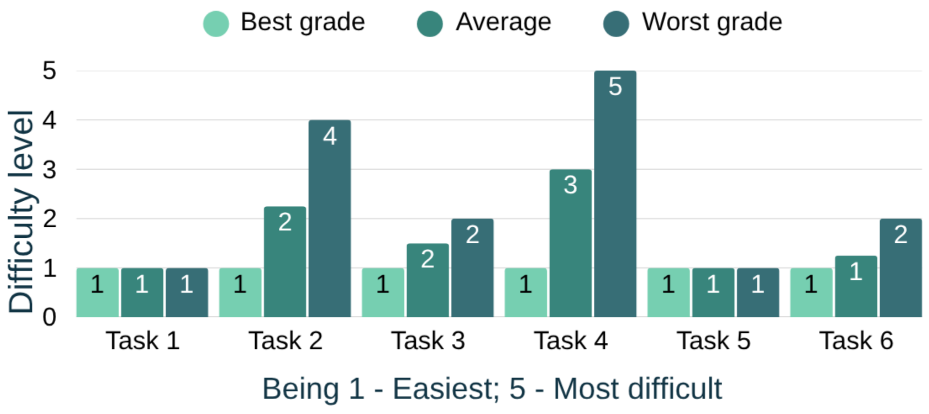


Figure 5.5: Perceived difficulty ratings for student tasks (best, average, worst).

Despite the small number of participants, this data was instrumental in identifying key usability strengths and areas for future improvement.

Observer Data Tables

The following tables summarize the observational data for each task, categorized by participant role (Student). Each table includes:

- Task completion status
- Average time taken
- Average number of errors
- Degree of confusion
- Whether help was requested
- Difficulty level observed (1 = Very easy, 5 = Very difficult)
- Additional notes

Table 5.3: Average time (min:sec) and average errors per task (n=4 participants)

Task	Average Time	Average Errors
Task 1	0:12	0.0
Task 2	0:55	0.3
Task 3	0:21	0.0
Task 4	1:24	0.3
Task 5	0:17	0.0
Task 6	0:41	0.5

Table 5.4: Average observed difficulty per task (1 = Very easy, 5 = Very difficult)

Task	Average Observed Difficulty
Task 1	1.0
Task 2	1.8
Task 3	1.5
Task 4	2.5
Task 5	1.0
Task 6	1.3

Table 5.5: Qualitative summary per task

Task	Observations
Task 1	Quick task, all participants completed without errors or confusion.
Task 2	Longer time and more errors, difficulties due to evaluation misconfiguration and submission issues.
Task 3	Easy, few errors, though some participants did not use filters, causing minor delays.
Task 4	Highest difficulty and time due to filtering challenges and missing MEC assignments, leading to confusion and errors.
Task 5	Very easy and fast; all identified the video correctly.
Task 6	Some errors and confusion related to hidden submit button, but no help was requested.

Task Success

In those tasks, participants were evaluated based on whether they achieved the correct results. As illustrated in Figure 5.6, the overall rate of task success, defined as reaching the correct conclusion, was 95.8%.

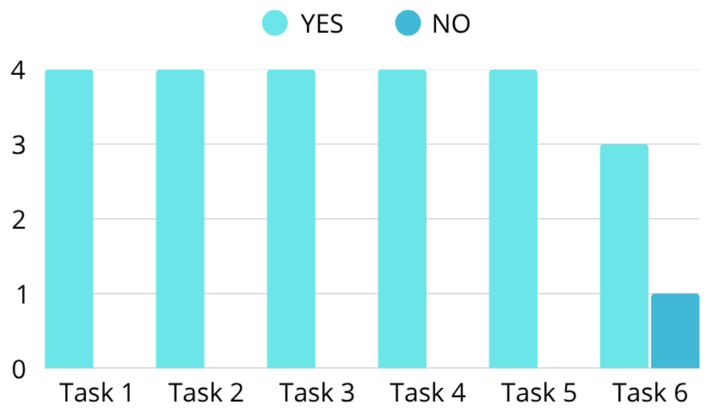


Figure 5.6: Tasks completed with correct results during usability tests.

SUS

Participants completed the SUS[11] questionnaire after the tasks. The median score per question was used to minimize bias. Positively worded questions (odd) and negatively worded ones (even) were scored inversely as per SUS guidelines.

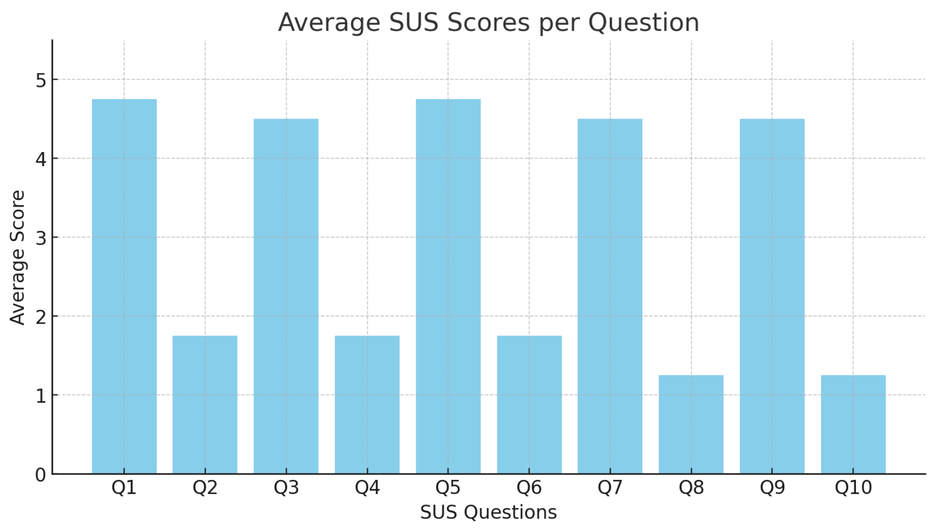
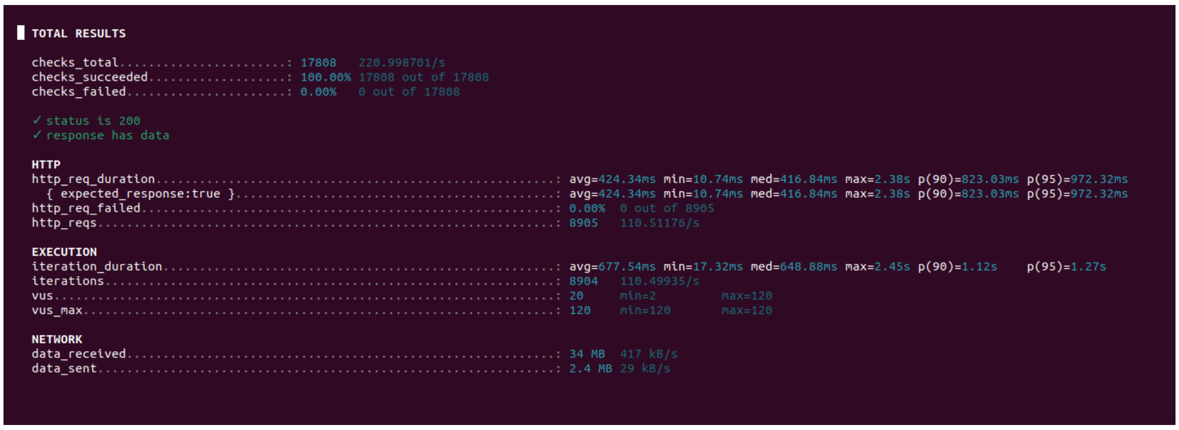


Figure 5.7: Median scores per question on the SUS.

The overall SUS score for the high-fidelity prototype was 91, an improvement from the previous fidelity version's score of 81, indicating enhanced usability. This value is well above the standard usability benchmark of 68, confirming excellent system usability.

5.3 Load Testing

To measure the API performance, load tests were conducted using the K6 tool [12]. Such testing is important to verify whether the application can handle a given number of users, i.e., if the application is scalable. The results obtained are summarized below.



```
█ TOTAL RESULTS

  checks_total.......................: 17808   220.998701/s
  checks_succeeded...................: 100.00% 17808 out of 17808
  checks_failed......................: 0.00%   0 out of 17808

  ✓ status is 200
  ✓ response has data

  HTTP
  http_req_duration.......................................................: avg=424.34ms min=10.74ms med=416.84ms max=2.38s p(90)=823.03ms p(95)=972.32ms
    { expected_response:true }............................................: avg=424.34ms min=10.74ms med=416.84ms max=2.38s p(90)=823.03ms p(95)=972.32ms
  http_req_failed.........................................................: 0.00%  0 out of 8905
  http_reqs...............................................................: 8905   110.51176/s

  EXECUTION
  iteration_duration......................................................: avg=677.54ms min=17.32ms med=648.88ms max=2.45s p(90)=1.12s    p(95)=1.27s
  iterations..............................................................: 8904   110.49935/s
  vus.....................................................................: 20     min=2         max=120
  vus_max.................................................................: 120    min=120       max=120

  NETWORK
  data_received...........................................................: 34 MB  417 kB/s
  data_sent...............................................................: 2.4 MB 29 kB/s
```

Figure 5.8: Summary of API performance during the load test using K6.

In this test, a total of 120 virtual users were simulated, with 50 concurrent connections sending requests to the API over a 30-second interval. The average response latency was **424.34 ms** and the average number of requests per second was **110.51**. The peak response latency of **2.38 s** occurred when the number of requests per second reached approximately **823**. The minimum latency recorded was **10.74 ms**, when the number of requests per second was around **191**.

Throughout the execution, a total of **8905 requests** were made, all of which succeeded with **0% failure rate**. The system sustained a maximum of **120 virtual users** without degradation in service.

Analyzing these results, we conclude that the API is capable of handling a high number of requests per second, well above the current real-world demand, while maintaining an acceptable and stable response time throughout the load.

5.4 Conclusion of Testing

Additionally, a third testing session was held on June 6th at the ESSUA, where 26 students (23 females, 3 males, ages ranging from 17 to 25) used the application simultaneously. This live scenario was designed to simulate peak usage and observe the platform's behavior under actual conditions. Thanks to the invaluable support of Professor Sílvia Badarra, who helped coordinate and gather all participants, the session was conducted smoothly.

Before starting, a detailed explanation was provided covering the overall concept, objectives of the session, and platform requirements to ensure clear understanding among all users. Throughout the session, the application remained completely stable and responsive, demonstrating excellent system stability under real-world load.

At the end of the session, all students completed a questionnaire based on the SUS to capture their feedback and overall opinion about the application's usability (see Appendix H for details).

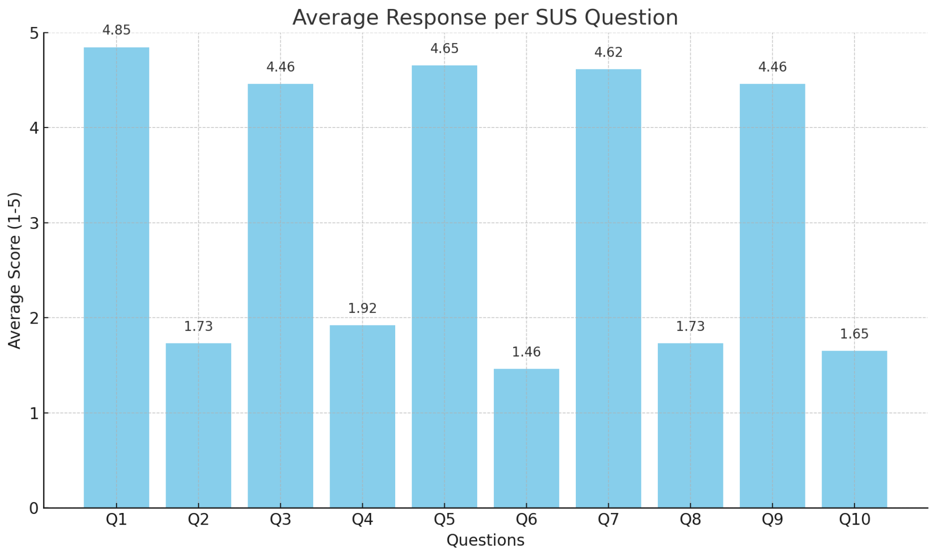


Figure 5.9: Average score per question in the SUS questionnaire.

As illustrated in Figure 5.9, the responses were consistently positive across most questions, particularly those related to ease of use, integration, and confidence, indicating a strong perceived usability. The obtained SUS score was **86**, which is a very high value, however this was still hindered by some evaluators that voted with 5 to everything, skewing our results - without these, our results could well be over 90. The fact that our sample was 26 students might have contributed to a high uncertainty in the sus scale.

Still, together with the controlled usability tests and load testing presented earlier, these results confirm that the application is both user-friendly and robust, capable of supporting expected user volumes while maintaining a high-quality experience.

CHAPTER 6

Conclusion

6.1 Project Overview

This project originated from the need presented by the Medicine Course at UA to have a digital way to perform their pratical evaluations at the Medicina Clinica course. It began by studying the current SoA, from which we concluded that many of the required features for the system were not met elsewhere, or the prices were too elevated for those solutions to be viable. As such, our task was to create a robust, complete solution that could be used by the university, and potentially other universities around the world, which was reflected in our architecture. Since our team was out of the loop in terms of what was necessary in the medicine course and what was needed, many meetings with our clients (highlighting meetings with Prof. Silvia Barreiro and Prof. Firmino Machado at this initial stage) were vital for us to get a headstart, as well as a initial Usability Test using a low-fidelity prototype, in order to get early feedback from both tutors and students. With that feedback, we were able to build a solution, with constant feedback from our advisor, as well as our clients, that could meet all the requests that were made to us at the project's inception. In the end, we performed new Usability Tests to verify that our solution was intuitive and ready to be used.

6.2 Main Results

Our system has not been used in a real scenario yet, but it is expected to be used next year in the Medicine courses. In the end, we were able to achieve all functional requirements, for a total of 43 requirements, as well as almost all non-functional requirements, for a total of 22 requirements. Additionally, the reception to our solution was extremely positive, from both our advisor, but especially from our clients, who are eager to be finally able to use our solution, having already scheduled meetings with other organizations and universities to discuss expanding this to other places. We can say that the result of this project has been overwhelmingly positive, and our goals have been achieved, all while meeting all deadlines proposed by ourselves.

6.3 Future Work

Our goals have been achieved, but we still need a bit more work until our solution is ready for the world.

- Our codebase is slightly disorganized, with filenames being ambiguous and the structure hard to navigate.
- Implement time based notifications, so that students can be reminded of an evaluation in the few days before it happens, so that they can prepare for it.
- We don't have tests, failing in the QA department. Implementing tests in our features allows for easier maintainability and detecting if any refactoring caused a functionality to break.
- We don't have a QA Manual, which would allow us to more smoothly ensure the quality of our solution.
- We don't follow GDPR guidelines. The solution to abide by these is detailed in Chatper 4, but to summarize, we would have to pseudo-anonymize personal data.
- Implement a LLM mechanism to provide feedback overview, to help students know what they need to improve

While these are important aspects to consider, we still believe that we focused on the right things in the implementation of this solution, though without some of these (mainly the GDPR guidelines) it is impossible to make this solution viable anywhere.

References

[1] C. D. Pereira, «Universidade de aveiro abre novo curso de medicina», *ET AL*, 2024, Acesso em 09 de junho de 2025. [Online]. Available: https://et-al.pt/2024/07/21/universidade-de-aveiro-abre-novo-curso-de-medicina/.

[2] O. Ten Cate, D. R. Taylor, C. Carraccio, S. Englander, J. Sherbino, and A. B. Holmboe, «Nuts and bolts of entrustable professional activities», *Journal of Graduate Medical Education*, vol. 5, no. 1, pp. 157–158, Mar. 2013. DOI: 10.4300/JGME-05-01-33. [Online]. Available: https://meridian.allenpress.com/jgme/article/5/1/157/200514/Nuts-and-Bolts-of-Entrustable-Professional.

[3] SIMPL Project. «SIMPL - supporting epa-based assessment». Accessed: 2025-06-14, University of Calgary. (2024), [Online]. Available: https://simpl.org/.

[4] EPA Portfolio. «EPA Portfolio - entrustable professional activities platform». Accessed: 2025-06-14, EPA Portfolio. (2024), [Online]. Available: https://epaportfolio.com/.

[5] EPA-Tracking-Tool. «Epa-tool – a platform for managing entrustable professional activities». Accessed: 2025-06-14, EPA-Tool. (2024), [Online]. Available: https://www.epatool.com/.

[6] Universidade de Aveiro. «Plataforma de elearning da universidade de aveiro». Acesso em 14 de junho de 2025, Universidade de Aveiro. (2025), [Online]. Available: https://elearning.ua.pt/.

[7] EvalMed, *EvalMed - YouTube Channel*, https://www.youtube.com/@EvalMed, Accessed: 2025-06-17, 2025.

[8] ELIXIR Europe, *ELIXIR: Interconnecting life science resources across Europe*, https://elixir-europe.org/, Accessed: 2025-06-14, 2024.

[9] G. Labs, *Spring boot observability dashboard for grafana*, Accessed: 2025-06-15, 2022. [Online]. Available: https://grafana.com/grafana/dashboards/17175-spring-boot-observability/.

[10] Figma. «Figma - collaborative interface design tool». Accessed: 2025-02-18. (2025), [Online]. Available: https://www.figma.com/.

[11] A. Bhat. «System usability scale: What it is, calculation + usage». Accessed: 2025-03-03, QuestionPro. (Aug. 2023), [Online]. Available: https://www.questionpro.com/blog/system-usability-scale/.

[12] Grafana Labs. «K6 – load testing for developers». Accessed: 2025-05-19, Grafana Labs. (2025), [Online]. Available: https://k6.io/.

APPENDIX A

Appendix A - API Documentation

EvalMed

Overview

This API exposes endpoints to manage EPAs and was designed to be used by medicine students and tutors from any university

Tags

Course

Endpoints for managing courses, including creation, retrieval, updating, and deletion of courses and their competencies.

Administrator

Endpoints for managing administrators, including CRUD operations and course management.

Reevaluation

Endpoints for managing reevaluation requests of EPAs for students, tutors, and administrators.

Tutor

This controller provides endpoints for managing tutors and their related activities, including student assignments, evaluations, feedback, and reevaluations. Only authenticated tutors are allowed to perform most actions.

Suggestions

Allows users to submit feedback or suggestions via email. Limits submissions to prevent spam.

Auth

Endpoints for authentication and user management, including OAuth token exchange, user retrieval, and one-time token generation.

Student

Endpoints for managing students, including creating, updating, deleting students, and retrieving their USF moments and final evaluations.

EPA

Endpoints for managing EPAs, nested EPAs, MECs, and evaluations.

Notification

Endpoints for managing user notifications and device tokens.

Person

Endpoints related to person management, including uploading profile images.

Feedback

Endpoints for students and tutors to create, respond to, and view feedback within a course.

Paths

GET `/api/v1/tutor/{id}` Get a tutor by ID

Fetches a tutor object by their ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/tutor/{id}` Update tutor information

Replaces all fields of a tutor with the provided information.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/tutor/{id}` Delete a tutor

Removes a tutor from the system by their ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/tutor/{id}` Patch tutor data

Updates only specific fields of a tutor.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/tutor/student/{studentId}/usf/{usfId}/evaluation` Replace evaluation for a student in a USF

Replaces all evaluation records for a specific student in a USF. Typically used when completely overwriting evaluations.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/tutor/student/{studentId}/usf/{usfId}/evaluation` Update evaluation for a student in a USF

Partially updates evaluations for a student in a USF. Requires tutor authentication or admin access.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/student/{studentId}/usf/{usfId}/self_evaluation` Replace self-evaluation for a student in a USF

Replaces all self-evaluation records for a specific student in a USF.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/student/{studentId}/usf/{usfId}/self_evaluation` Update self-evaluation for a student in a USF

Updates specific fields of the self-evaluation for a student in a USF.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	**Description**	**Links**
200	OK	No Links

GET `/api/v1/student/{id}` Get a student by ID

Fetches a student object by their ID.

Parameters

Type	**Name**	**Description**	**Schema**
path	**id** *required*		

Responses

Code	**Description**	**Links**
200	OK	No Links

PUT `/api/v1/student/{id}` Update a student

Replaces all fields of a student with the provided information.

Parameters

Type	**Name**	**Description**	**Schema**
path	**id** *required*		

Responses

Code	**Description**	**Links**
200	OK	No Links

DELETE `/api/v1/student/{id}` Delete a student

Removes a student from the system by their ID.

Parameters

Type	**Name**	**Description**	**Schema**
path	**id** *required*		

Responses

Code	**Description**	**Links**
200	OK	No Links

PATCH `/api/v1/student/{id}` Patch student data

Updates only specific fields of a student.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/reevaluation/{id}/reply` Reply to a reevaluation

Allows tutors to reply to reevaluations with a state update. Only accessible to tutors.

Parameters

Type	Name	Description	Schema
path	**id** *required*		
query	**state** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/notifications/{id}` Get notification by ID

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/notifications/{id}` Update a notification

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/notifications/{id}` Delete a notification

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/notifications/{id}` Patch a notification

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/feedback/{id}` Get feedback for a specific student ID

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/feedback/{id}` Respond to a feedback request

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/{id}` Get all EPAs for a specific university and course

Retrieves all EPAs associated with a given university and course.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/epa/{id}` Update a certain EPA

Updates the details of an existing EPA based on its ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/epa/{id}` Delete a certain EPA

Deletes an existing EPA based on its ID. Requires admin access to the course.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/epa/admin/mec/{epaId}/{epaNestedId}/{mecId}` Update MEC details

Updates the details of an existing MEC based on its ID.

Parameters

Type	Name	Description	Schema
path	**epaId** *required*		
path	**epaNestedId** *required*		
path	**mecId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/course/{university_id}/{course_id}` Get a course by ID

Fetches a course object by its ID within a specific university.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/course/{university_id}/{course_id}` Update a certain course in an university

Updates an existing course in a specific university and returns the updated course object.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/course/{university_id}/{course_id}` Delete a course

Removes a course from the system by its ID within a specific university.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/course/{university_id}/{course_id}` Patch a certain course in an university

Updates specific fields of an existing course in a specific university and returns the updated course object.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/administrator/{universityId}/{courseId}/{administratorId}` Update an administrator for a course

Updates an existing administrator's details for a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		

Type	Name	Description	Schema
path	**administratorId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/administrator/{universityId}/{courseId}/{administratorId}` Delete an administrator from a course

Deletes an administrator from a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**administratorId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/{id}` Get all administrators

Retrieves a list of all administrators, regardless of their enabled status.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/administrator/{id}` Update an administrator

Updates an existing administrator and returns the updated administrator object.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/administrator/{id}` Delete an administrator

Deletes an administrator by their ID and returns a success response.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/administrator/{id}` Patch an administrator

Updates specific fields of an existing administrator and returns the updated administrator object.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/administrator/universities/{universityId}/courses/{courseId}/tutors/{tutorId}` Update tutor details

Updates the details of a tutor for a specific course and university.

Parameters

Type	Name	Description	Schema
path	**tutorId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/administrator/universities/{universityId}/courses/{courseId}/tutors/{tutorId}` Delete a tutor from a course

Deletes a tutor from a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**tutorId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/administrator/universities/{universityId}/courses/{courseId}/students/{studentId}` Update student details

Updates the details of a student for a specific course and university.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/administrator/universities/{universityId}/courses/{courseId}/students/{studentId}` Get all students and tutors for a course

Retrieves a list of all students and tutors associated with a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/administrator/universities/{universityId}/courses/{courseId}/students/{studentId}/tutors/{tutorId}` Update student and tutor

Updates the details of a student and their assigned tutor for a specific course and university.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**universityId** *required*		
path	**tutorId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/universities/{universityId}/courses/{courseId}/students/{studentId}/tutors/{tutorId}` Assign a tutor to a student

Assigns a tutor to a student for a specific course and university. Requires admin access.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**tutorId** *required*		

Type	Name	Description	Schema
query	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/administrator/universities/{universityId}/courses/{courseId}/students/{studentId}/tutors/{tutorId}` Unassign a tutor from a student

Unassigns a tutor from a student for a specific course and university. Requires admin access.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**tutorId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/course/{universityId}/{courseId}/usf/moments` Get USF moments for a course

Retrieves a list of USF moments associated with a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PUT `/api/v1/administrator/course/{universityId}/{courseId}/usf/moments` Update USF moments for a course

Updates the list of USF moments for a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/tutor/{id}/reevaluation` Get tutor reevaluations

Returns a list of reevaluations associated with the tutor. Authentication as the tutor is required.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/tutor/{id}/reevaluation` Create reevaluation

Creates a new reevaluation request for a student by a tutor. Authentication as the tutor is required.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/tutor/{id}/reevaluation` Patch reevaluation

Updates specific fields of an existing reevaluation request.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/tutor/{id}/feedback` Create feedback

Creates a new feedback entry for a student by a tutor. Authentication as the tutor is required.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/tutor/{id}/feedback` Patch feedback

Updates specific fields of an existing feedback entry.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/tutor/bulk` Create tutors in bulk

Adds multiple tutors at once to the system.

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/tutor/` Get all enabled tutors

Retrieves a list of all currently enabled tutors.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/tutor/` Create a new tutor

Creates and stores a new tutor in the system.

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{id}/final` Get the final evaluation for a student

Retrieves the final evaluation for a specific student by their ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/student/{id}/final` Create a final evaluation for a student

Creates a final evaluation for a specific student by their ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/student/bulk` Bulk create students

Creates multiple students in a single request.

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/` Get all enabled students

Retrieves a list of all currently enabled students.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/student/` Create a new student

Creates a new student and returns the created student object.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/reevaluation/{universityId}/{courseId}` Create reevaluations for a course

Allows students to request reevaluations for EPAs in a specific course, requiring authentication as a student.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/notifications/type/{userType}` Get notifications by user type

Parameters

Type	Name	Description	Schema
path	**userType** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/notifications/type/{userType}` Create a notification for all users of a type

Parameters

Type	Name	Description	Schema
path	**userType** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/notifications/` Get all notifications

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/notifications/` Create a notification for a specific user

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/feedback/{universityId}/{courseId}` Create a feedback request

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/epa/{universityId}/{courseId}` Get all EPAs for a specific university and course

Retrieves all EPAs associated with a given university and course.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/epa/{epaId}/nested` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**epaId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/epa/student/{studentId}/usf/{usfId}` Submit an evaluation for a student in a USF

Allows students or tutors to submit evaluations for a specific student in a USF.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/epa/student/{studentId}/usf/{usfId}` Edit an existing evaluation for a student in a USF

Allows students or tutors to edit evaluations for a specific student in a USF.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/epa/mec` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/epa/admin/mec` Create a new MEC

Creates a new MEC with details for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
query	**epaId** *required*		
query	**epaNestedId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/course/` Get all enabled courses

Retrieves a list of all currently enabled courses.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/course/` Create a new course

Creates a new course in the system and returns the created course object.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/{universityId}/{previousCourseId}/new/year` Create a new course and USF moments

Creates a new course and associated USF moments based on the provided data.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**previousCourseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/{universityId}/{courseId}` Create an administrator for a course

Creates a new administrator and assigns them to a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/{universityId}/{courseId}/tutor` Create a tutor for a course

Creates a new tutor and assigns them to a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/{universityId}/{courseId}/student` Get all students and tutors for a course

Retrieves a list of all students and tutors associated with a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/bulk-import` Bulk import data for a course

Imports data in bulk for a specific course and university.

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/` Get all enabled administrators

Retrieves a list of all currently enabled administrators.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/v1/administrator/` Create a new administrator

Creates a new administrator and returns the created administrator object.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/suggestions` Submit a suggestion

Allows a user to submit a suggestion message. Submission is limited to once every 5 hours per user.

Parameters

Type	Name	Description	Schema
query	**userName** *required*		

Responses

Code	Description	Links
200	Suggestion submitted successfully	No Links
400	Missing or blank suggestion message	No Links
429	Suggestion rate limit exceeded	No Links
500	Internal server error while sending email	No Links

POST `/api/person/{id}/upload-profile-image` Upload profile image for a person

Allows a student to upload a profile image. Maximum size is 2MB. Replaces any existing image.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	Image uploaded successfully	No Links
400	Image too large or invalid	No Links
403	Access denied	No Links
404	Person not found	No Links
500	Server error while saving image	No Links

POST `/api/auth/token` Exchange authorization code for access token

Exchanges the authorization code received from the OAuth provider for an access token and user information.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/auth/refresh` Refresh access token

Refreshes the access token using the refresh token stored in cookies.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/auth/ott` Generate and send one-time token (OTT)

Generates a one-time token (OTT) for the specified email and sends it to the user. The email must be registered in the system.

Parameters

Type	Name	Description	Schema
query	**email** *required*		

Responses

Code	Description	Links
200	OK	No Links

POST `/api/auth/ott/token` Trade OTT for token

Trades a one-time token (OTT) for a user token. The OTT must be provided in the request body.

Responses

Code	Description	Links
200	OK	No Links

POST `/api/auth/logout` Logout

Logs out the user by clearing the access and refresh tokens stored in cookies.

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/notifications/remove-token` Remove FCM token from current user

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/notifications/clear-tokens` Clear all FCM tokens from current user

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/notifications/add-token` Add FCM token to current user

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/epa/{id}/name` Patch a certain EPA

Partially updates an existing EPA based on its ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/epa/{epaId}/nested/{nestedId}` Delete a nested EPA

Deletes a nested EPA by its ID from a specific EPA

Parameters

Type	Name	Description	Schema
path	**epaId** *required*		
path	**nestedId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/epa/{epaId}/nested/{nestedId}` Get all Nested EPAs for a specific EPA

Retrieves all nested EPAs associated with a given EPA ID.

Parameters

Type	Name	Description	Schema
path	**epaId** *required*		
path	**nestedId** *required*		

Responses

Code	Description	Links
200	OK	No Links

PATCH `/api/v1/epa/mec/student/{studentId}/usf/{usfId}` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		

Type	Name	Description	Schema
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/tutor/{id}/students` Get students assigned to a tutor

Returns the list of students assigned to a given tutor. The request must be authenticated as the tutor.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{universityId}/{courseId}/{studentId}/usf-progress` Get USF progress for a student

Retrieves the USF progress data for a specific student by their ID.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{studentId}/usf/{usfId}` Get a specific USF moment for a student

Retrieves a specific USF moment for a student by their ID and USF moment ID.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{studentId}/usf/{usfId}/mecs` Get MECs for a student’s USF moment

Retrieves all MECs for a specific student’s USF moment.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{id}/courses` Get courses for a student

Retrieves all courses associated with a specific student by their ID.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{courseId}/{universityId}/{studentId}/usf` Get USF moments for a student

Retrieves all USF moments for a specific student.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{courseId}/{universityId}/{id}/usf/mecs` Get USF moments with MECs for a student

Retrieves all USF moments with MECs for a specific student.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**id** *required*		

Type	Name	Description	Schema
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{courseId}/{universityId}/{id}/next` Get next evaluation with MECs for a student

Retrieves the next evaluation with MECs for a specific student by their ID.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**id** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/student/{courseId}/{universityId}/{id}/last` Get last evaluation with MECs for a student

Retrieves the last evaluation with MECs for a specific student by their ID.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**id** *required*		

Type	Name	Description	Schema
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/reevaluation` Get all reevaluations for the authenticated user

Retrieves reevaluations based on the user type: students, tutors, or admins.

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/reevaluation/{id}` Get all reevaluations for a specific person

Fetches reevaluations for a person by their ID, accessible to students, tutors, and admins.

Parameters

Type	Name	Description	Schema
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/notifications/user/{userId}` Get notifications by user ID

Parameters

Type	Name	Description	Schema
path	**userId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/notifications/me` Get notifications for the current user

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/notifications/me/count` Get unread notification count for the current user

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/feedback` Get feedback for the current student or tutor

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/feedback/timeline/{universityId}/{courseId}` Get feedback timeline data for course

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/{university_id}/{course_id}/{student_id}/mec` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		
path	**student_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/{universityId}/{courseId}/structure` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/{id}/nested/{nestedId}` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**nestedId** *required*		
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/{id}/nested/{nestedEPAId}/mec/{mecId}` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**mecId** *required*		
path	**nestedEPAId** *required*		
path	**id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/student/{studentId}/usf/{usfId}/evaluation` Get evaluation details for a student in a USF

Fetches detailed evaluation information for a specific student in a USF.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**usfId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/name/{name}` Get EPA by name

Retrieves an EPA by its name. Returns null if not found.

Parameters

Type	Name	Description	Schema
path	**name** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/mec/{epaId}/{epaNestedId}/{mecId}` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**epaId** *required*		
path	**epaNestedId** *required*		
path	**mecId** *required*		

Responses

Code	Description	Links
200	OK	No Links

DELETE `/api/v1/epa/mec/{epaId}/{epaNestedId}/{mecId}` Delete a MEC

Deletes an existing MEC by its ID from a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**epaId** *required*		
path	**epaNestedId** *required*		
path	**mecId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/mec/{epaId}/{epaNestedId}/{mecId}/student/{studentId}` Get MEC completion statistics

Fetches MEC completion statistics for a specific EPA and nested EPA.

Parameters

Type	Name	Description	Schema
path	**studentId** *required*		
path	**epaId** *required*		
path	**epaNestedId** *required*		
path	**mecId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/course/{courseId}/{universityId}/mec-completion-statistics` Get MEC completion statistics

Fetches MEC completion statistics for a specific course and university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/epa/` Get all enabled EPAs

Fetches all EPAs that are currently enabled in the system.

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/course/{university_id}` Get all courses in a university

Retrieves a list of all courses in a specific university.

Parameters

Type	Name	Description	Schema
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/course/{university_id}/{course_id}/competencies` Get competencies for a course

Retrieves a list of competencies associated with a specific course in a university.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/{university_id}/{course_id}` Get administrators by university and course

Retrieves a list of administrators associated with a specific university and course.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/course/{university_id}/{course_id}/students` Get students by course

Retrieves a list of students enrolled in a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/course/{universityId}/{courseId}/usf/stats` Get tutors by course

Retrieves a list of tutors associated with a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/course/{universityId}/{courseId}/available/tutors` Get available tutors for a course

Retrieves a list of available tutors for a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		

Type	Name	Description	Schema
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/course/planned/{universityId}/{courseId}` Get planned courses for a specific course and university

Retrieves a list of planned courses for a specific course at a university.

Parameters

Type	Name	Description	Schema
path	**universityId** *required*		
path	**courseId** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/v1/administrator/course/overview/{university_id}/{course_id}` Get course overview

Retrieves an overview of a specific course, including details such as students, tutors, and USF moments.

Parameters

Type	Name	Description	Schema
path	**course_id** *required*		
path	**university_id** *required*		

Responses

Code	Description	Links
200	OK	No Links

GET `/api/auth/user/me` Get current authenticated user

Returns the current authenticated user’s data.

Responses

Code	Description	Links
200	OK	No Links

Components

APPENDIX B

Appendix B - Student Task Form (Low Fidelity Prototype)

Testes de usabilidade Aluno - Tarefas EvalMed

Bem vindo aos testes de usabilidade para a aplicação de gestão de EPAs, EvalMed.

O objetivo destes testes são utilizar a aplicação realizar as tarefas indicadas em baixo.
Pedimos que tente ser o mais honesto possível, caso alguma tarefa seja mais difícil ou confusa é bom que manifeste tal dificuldade para podermos melhorar.
Pode em qualquer momento pedir ajuda ao observador, mas apenas em último recurso.
Sinta-se à vontade para explorar a aplicação e encontrar o caminho até ao objetivo final.

* Indica uma pergunta obrigatória

1. Nº participante *

Tarefa 1

Realize a auto avaliação da próxima avaliação (USF) com os seguintes níveis de autonomia:
MEC14 - N1
MEC15 - N3
MEC16 - N3

2. Qual a resposta obtida após submeter a auto avaliação? *

3. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

4. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 2

Verifique qual a avaliação obtida na USF4.

5. Qual foi o nível de autonomia alcançado no MEC 3? *

6. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

7. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 3

Ver a quantidade de pontos de MECs aprovados.

8. Qual o valor de pontos aprovados? *

9. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

10. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 4

Ver procedimento do MEC 1.

11. Quantas etapas tem este MEC? *

12. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

13. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 5

Peça reavaliação do MEC 3.

14. Qual o feedback obtido? *

15. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

16. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

APPENDIX C

Appendix C - Tutor Task Form (Low Fidelity Prototype)

Testes de usabilidade Tutor - Tarefas EvalMed

Bem vindo aos testes de usabilidade para a aplicação de gestão de EPAs, EvalMed.

O objetivo destes testes são utilizar a aplicação realizar as tarefas indicadas em baixo. Pedimos que tente ser o mais honesto possível, caso alguma tarefa seja mais difícil ou confusa é bom que manifeste tal dificuldade para podermos melhorar.
Pode em qualquer momento pedir ajuda ao observador, mas apenas em último recurso. Sinta-se à vontade para explorar a aplicação e encontrar o caminho até ao objetivo final.

* Indica uma pergunta obrigatória

1. Nº participante *

Tarefa 1

Verifique quando é a próxima avaliação

2. Qual o dia e hora da próxima avaliação? *

3. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	○	○	○	○	○	Muito difícil

4. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 2

Realize avaliação do USF5 para a Maria do Mar com os seguintes níveis de autonomia:
MEC14 - N1
MEC15 - N2
MEC16 - N2

5. Qual é o feedback obtido após submeter a avaliação? *

6. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

7. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 3

Verifique o calendário.

8. Em que dia é a próxima avaliação? *

9. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

10. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 4

Ver procedimento do MEC 1.

11. Quantas etapas tem este MEC? *

12. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

13. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 5

Aceite o pedido de um aluno para reavaliar MECs no próximo USF.

14. Qual o feedback obtido? *

15. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

16. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

APPENDIX D

Appendix D - Observer Form (Low Fidelity Prototype)

Formulário Observador

Um formulário para o observador preencher enquanto vê o participante a realizar a tarefa

* Indica uma pergunta obrigatória

1. Nº Participante *

2. Participante *

Marcar apenas uma oval.

◯ Tutor

◯ Aluno

Tarefa 1

Tutor - **Identifica a data da próxima avaliação** (07/05/2025, Home ou Calendário)
Aluno - **Realiza a próxima autoavaliação com parâmetros específicos** (

- MEC14 - N1
- MEC15 - N3
- MEC16 - N3

(Tem a certeza que deseja submeter esta autoavaliação? Não será possível editá-la depois de submetida.)

3. O participante completou a tarefa? *

Marcar apenas uma oval.

- [ ] Não
- [ ] Sim

4. Tempo observado (máximo: 05:00) *

5. Número de erros *

6. O participante esta perdido? *

Marcar apenas uma oval.

- [ ] não
- [ ] um pouco
- [ ] muito

7. Pediu ajuda? *

Marcar apenas uma oval.

- [ ] Não
- [ ] Sim

8. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

9. Observações extra

Tarefa 2

Tutor -
Realiza a avaliação da Maria do Mar com parâmetros específicos (

- MEC14 - N1
- MEC15 - N3
- MEC16 - N3

(Tem a certeza que deseja submeter esta avaliação? Não será possível editá-la depois de submetida.)
Aluno **- Verifica a nota da avaliação obtida na USF4 (N2)**

10. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

11. Tempo observado (máximo: 05:00) *

12. Número de erros *

13. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

14. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

15. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

16. Observações extra

Tarefa 3

Tutor -
Verifique o calendário.

(21/05/2025)
Aluno - Ver a quantidade de pontos de MECs aprovados.

(11)

17. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

18. Tempo observado (máximo: 05:00) *

19. Número de erros *

20. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

21. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

22. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

23. Observações extra

Tarefa 4

Tutor - Ver procedimento do MEC 1**(4)**
Aluno - Ver procedimento do MEC 1**(4)**

24. O participante completou a tarefa? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim

25. Tempo observado (máximo: 05:00) *

26. Número de erros *

27. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

28. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

29. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

30. Observações extra

Tarefa 5

Tutor - **Aceite o pedido de um aluno para reavaliar MECs no próximo USF.** (aceite)
Aluno - Peça reavaliação do MEC 3. (
Deseja submeter o pedido de reavaliação?/ A aguadar)

31. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

32. Tempo observado (máximo: 05:00) *

33. Número de erros *

34. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

35. Pediu ajuda? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim

36. Dificuldade observada *

Marcar apenas uma oval.

- ◯ 1 - Muito fácil
- ◯ 2 - Fácil
- ◯ 3 - Razoável
- ◯ 4 - Difícil
- ◯ 5 - Muito difícil

37. Observações extra

APPENDIX E

Appendix E - Post-Task Questionnaire (Low Fidelity Prototype)

Formulário pós-tarefas

Instruções: Obrigado pela cooperação nesta experiência, cujo objetivo era avaliar a interface de utilizador da nossa aplicação e tentar melhorar a mesma.
A sua colaboração é crucial para o sucesso do nosso desenvolvimento e vai certamente influenciar o nosso trabalho.
Pedimos então que preencha este último formulário, cujos dados vão ser usado única e exclusivamente para motivos de estudo futuro.

* Indica uma pergunta obrigatória

Dados demográficos

1. Nº de participante *

2. Género *

Marcar apenas uma oval.

- ◯ Masculino
- ◯ Feminino
- ◯ Outra: ____________

3. Idade *

4. Situação profissional *

Marcar apenas uma oval.

- Estudante
- Médico/Tutor

5. Observações (quaisquer factos relevantes que afetem a performance no teste, ex.: problemas de visão, motores, ...)

Opinião geral sobre o sistema

Depois de usar o nosso sistema e concluir as tarefas, escolha a opção que melhor reflita a sua opinião sobre o mesmo. Se não achar adequado nenhuma das opções, escolha **3**

6. Vejo-me a usar este serviço futuramente em detrimento do sistema atual *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc						Concordo totalmente

7. Achei o sistema desnecessariamente complexo *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

8. Achei a aplicação fácil de usar *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

9. Acho que precisaria de algum apoio técnico para usar este sistema *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

10. Achei que as várias funcionalidades do sistema estavam bem integradas *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

11. Achei que o sistema estava demasiado inconsistente

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo Totalmente

12. Acho que a maioria das pessoas aprenderia rápido a usar este sistema *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

13. Achei a aplicação complicada de usar *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

14. Senti-me confiante ao navegar na aplicação *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

15. Precisei de aprender muitas cosias antes de começar a usar a aplicação *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

16. Comentários extra sobre a experiência:

Em nome da equipa **EvalMed**, muito obrigado pela sua colaboração!

APPENDIX F

Appendix F - Student Task Form (High Fidelity Prototype)

Testes de usabilidade Aluno - Tarefas EvalMed

Bem vindo aos testes de usabilidade para a aplicação de gestão de EPAs, EvalMed.

O objetivo destes testes são utilizar a aplicação realizar as tarefas indicadas em baixo.
Pedimos que tente ser o mais honesto possível, caso alguma tarefa seja mais difícil ou confusa é bom que manifeste tal dificuldade para podermos melhorar.
Pode em qualquer momento pedir ajuda ao observador, mas apenas em último recurso.
Sinta-se à vontade para explorar a aplicação e encontrar o caminho até ao objetivo final.

* Indica uma pergunta obrigatória

1. Nº participante *

Tarefa 1

Identifica a data da próxima avaliação

2. Qual é a data da próxima avaliação? *

Exemplo: 7 de janeiro de 2019

3. Em que página conseguiste encontrar a data da próxima avaliação? *

4. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	○	○	○	○	○	Muito difícil

5. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 2

Realize a auto avaliação da próxima avaliação (USF) com níveis de autonomia à tua escolha:

Exemplo de preenchimento
MEC1 - N1
MEC2 - N3
MEC3 - N3

6. Qual a resposta que o sistema te deu logo após clicares em "Submeter Avaliação"? *

7. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

8. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 3

Verifique qual a avaliação obtida na USF4

9. Qual foi o nível de autonomia alcançado no 1º MEC avaliado? *

10. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

11. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 4

Identifica qual o 1º MEC que faz parte do EPA "Comunicar e colaborar com outros profissionais" e está Aprovado

12. Qual o MEC que corresponde aos parâmetros de pesquisa? *

13. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

14. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 5

Ver procedimento e vídeo do Marco Educativo Clínico (MEC) 1.2.3

15. Quantas etapas tem este MEC? *

16. No vídeo explicativo do procedimento, após clicares no link do vídeo, qual era o nome do Vídeo? *

17. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

18. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

Tarefa 6

Solicite reavaliação do Marco Educativo Clínico (MEC) com possível melhoria.

19. Qual a resposta que o sistema te deu logo após clicares em "Confirmar"? *

20. Dificuldade *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito difícil

21. Comentários adicionais (se não conseguiste completar a tarefa explica porquê)

APPENDIX G

Appendix G - Observer Form (High Fidelity Prototype)

Formulário Observador

Um formulário para o observador preencher enquanto vê o participante a realizar a tarefa

* Indica uma pergunta obrigatória

1. Nº Participante *

2. Participante *

   Marcar apenas uma oval.

   ◯ Tutor

   ◯ Aluno

Tarefa 1

Tutor - **Identifica a data da próxima avaliação** (07/05/2025, Home ou Calendário)
Aluno - **Identifica a data da próxima avaliação** (07/05/2025, Home ou Calendário)
Administrador - **Qual é a média atual ()**

3. O participante completou a tarefa? *

   Marcar apenas uma oval.

   ◯ Não

   ◯ Sim

4. Tempo observado (máximo: 05:00) *

5. Número de erros *

6. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

7. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

8. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

9. Observações extra

Tarefa 2

Tutor - **Realize a avaliação do USF próximo para um aluno da cadeira de Medicina Clínica I com avaliações à tua escolha. (Tem a certeza que deseja submeter esta avaliação? Não será possível editá-la depois de submetida.)**
Aluno - **Realize a auto avaliação da próxima avaliação (USF) com níveis de autonomia à tua escolha: (Tem a certeza que deseja submeter esta avaliação? Não será possível editá-la depois de submetida.)**
Administrador - **Atribua um tutor disponível a um aluno que ainda não tenha tutor.**

10. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

11. Tempo observado (máximo: 05:00) *

12. Número de erros *

13. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

14. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

15. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

16. Observações extra

Tarefa 3

Tutor - **Responde a um pedido de feedback enviados por um aluno ("Resposta enviada com sucesso!")**
Aluno - **Verifique qual a avaliação obtida na USF4 realizada a 20/11/2024 (N1)**
Administrador -**Altera a hora da avaliação do dia 21/05/2025 para a hora de início 10:00h e a hora final 14:00h ("Momentos de avaliação atualizadas com sucesso!")**

17. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

18. Tempo observado (máximo: 05:00) *

19. Número de erros *

20. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

21. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

22. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

23. Observações extra

Tarefa 4

Tutor -**Aceita um pedido de reavaliação de um MEC para a próxima USF, solicitado por um aluno (Aceite)**
Aluno - **Identifica qual o MEC que faz parte do EPA "Comunicar e colaborar com outros profissionais", tem nível Previsto de autonomia N1 e está Por Avaliar (** Procedimento de pedido de informação clínica)
Administrador - **Programa o próximo ano letivo com apenas uma avaliação. Podes preencher os campos com valores escolhidos por ti. ("Novo ano planeado com sucesso!")**

24. O participante completou a tarefa? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim

25. Tempo observado (máximo: 05:00) *

____

26. Número de erros *

____

27. O participante esta perdido? *

Marcar apenas uma oval.

- ◯ não
- ◯ um pouco
- ◯ muito

28. Pediu ajuda? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim

29. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

30. Observações extra

Tarefa 5

Tutor - **Identifica qual o MEC que faz parte do EPA "Comunicar e colaborar com outros profissionais", tem nível Previsto de autonomia N1 e está "Por Avaliar"** (Procedimento de pedido de informação clínica)
Aluno - **Ver procedimento e vídeo do Marco Educativo Clínico (MEC) 1.2.3 (3,**
Como medir a tensão arterial?
)
Administrador - **Visualiza os dados de um estudante (** ponte@ua.pt, 125853, Matilde Pinto)

31. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

32. Tempo observado (máximo: 05:00) *

33. Número de erros *

34. O participante esta perdido? *

Marcar apenas uma oval.

- ◯ não
- ◯ um pouco
- ◯ muito

35. Pediu ajuda? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim

36. Dificuldade observada *

Marcar apenas uma oval.

- ◯ 1 - Muito fácil
- ◯ 2 - Fácil
- ◯ 3 - Razoável
- ◯ 4 - Difícil
- ◯ 5 - Muito difícil

37. Observações extra

Tarefa 7

Tutor - **Enviar sugestão para a nossa equipa de suporte com a mensagem "Sucesso" (Obrigada pela tua sugestão!)**
Aluno - **Solicite reavaliação do Marco Educativo Clínico (MEC) com possível melhoria. (Reavaliações submetidas com sucesso!)**
Administrador - **Edita um Marco Educativo clínico(MEC) à tua escolha ("Mec atualizado com sucesso!")**

38. O participante completou a tarefa? *

Marcar apenas uma oval.

- Não
- Sim

39. Tempo observado (máximo: 05:00) *

40. Número de erros *

41. O participante esta perdido? *

Marcar apenas uma oval.

- não
- um pouco
- muito

42. Pediu ajuda? *

Marcar apenas uma oval.

- Não
- Sim

43. Dificuldade observada *

Marcar apenas uma oval.

- 1 - Muito fácil
- 2 - Fácil
- 3 - Razoável
- 4 - Difícil
- 5 - Muito difícil

44. Observações extra

APPENDIX H

Appendix H - SUS Questionnaire (Final Testing)

Formulário SUS

Instruções: Obrigado pela cooperação nesta experiência, cujo objetivo era avaliar a interface de utilizador da nossa aplicação e tentar melhorar a mesma.
A sua colaboração é crucial para o sucesso do nosso desenvolvimento e vai certamente influenciar o nosso trabalho.
Pedimos então que preencha este último formulário, cujos dados vão ser usado única e exclusivamente para motivos de estudo futuro.

* Indica uma pergunta obrigatória

Dados demográficos

1. Género *

Marcar apenas uma oval.

- ◯ Masculino
- ◯ Feminino
- ◯ Outra: ____________

2. Idade *

____

3. Situação profissional *

Marcar apenas uma oval.

- ◯ Estudante
- ◯ Médico/Tutor

4. Observações (quaisquer factos relevantes que afetem a performance no teste, ex.: problemas de visão, motores, ...)

Opinião geral sobre o sistema

Depois de usar o nosso sistema e concluir as tarefas, escolha a opção que melhor reflita a sua opinião sobre o mesmo. Se não achar adequado nenhuma das opções, escolha **3**

5. Vejo-me a usar este serviço futuramente em detrimento do sistema atual *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

6. Achei o sistema desnecessariamente complexo *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

7. Achei a aplicação fácil de usar *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

8. Acho que precisaria de algum apoio técnico para usar este sistema *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

9. Achei que as várias funcionalidades do sistema estavam bem integradas *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

10. Achei que o sistema estava demasiado inconsistente

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo Totalmente

11. Acho que a maioria das pessoas aprenderia rápido a usar este sistema *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

12. Achei a aplicação complicada de usar *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

13. Senti-me confiante ao navegar na aplicação *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

14. Precisei de aprender muitas cosias antes de começar a usar a aplicação *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

15. A aplicação foi rápida a carregar as páginas?

Marcar apenas uma oval.

	1	2	3	4	5	6	7	8	9	10	
Muit	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Lento

16. Comentários extra sobre a experiência:

Em nome da equipa **EvalMed**, muito obrigado pela sua colaboração!

APPENDIX I

Appendix I - Personal Data Consent Form

Consentimento Informado

EvalMed

**Coordenadores: Pedro Ponte, Ricardo Antunes,
Afonso Ferreira, Tomás Brás, Carolina Silva**

Procedimento

Os participantes vão realizar um conjunto de tarefas previamente definidas usando uma aplicação web para gerir a avaliação das suas EPAs. Durante o procedimento vão ser recolhidos dados sobre o seu perfil demográfico e comentários ou dificuldades aquando da realização das tarefas e no uso geral da aplicação.

Duração

Esta experiência deve demorar entre 10 a 20 minutos.

Riscos para o participante

Não haverá riscos para o participante.

Benefícios para o participante

Os participantes podem influenciar diretamente uma aplicação que será usada futuramente pelos mesmos.

Confidencialidade

Todos os dados recolhidos durante a experiência são anónimos e confidenciais e apenas serão usados neste formulário para análise e discussão no futuro desenvolvimento da aplicação.

Participação voluntária

A sua participação é completamente voluntária. Mesmo aceitando participar, pode a qualquer momento desistir, manifestando tal vontade com o observador. Nesse caso, todos os dados recolhidos até ao momento são descartados.

Contactos

Para qualquer questão relacionada com esta experiência, contactar qualquer um dos coordenadores: Pedro Ponte (ponte@ua.pt) , Ricardo Antunes (ricardo.alexandre.antunes@ua.pt), Afonso Ferreira (afonso.ferreira@ua.pt), Tomás Brás (tomasbras@ua.pt), Carolina Silva (carolinapsilva@ua.pt)

Data: __/03/2025

Por favor insira o seu nome: X

Por favor assine: X

Muito obrigado!

Agradecemos a sua participação!
A equipa,
EvalMed

APPENDIX J

Appendix J - Image Consent Form

Consentimento Imagem
EvalMed

Eu, ______________________________, autorizo a captação da minha imagem durante os testes de usabilidade da **EvalMed** através de fotos e/ou vídeos e a sua utilização futura para fins promocionais do projeto.
Entendi que não tenho direito a qualquer renumeração resultante do uso das imagens e que a qualquer momento posso pedir para as mesmas não serem utilizadas em situações futuras.

Data: __/__/____

Assinatura: