Visualising Data in NAPP Web Application

Catarina Alexandra Pereira Cepeda

Thesis to obtain the Master of Science Degree in Information Systems and Computer Engineering

Supervisors: Prof. Ana Isabel Baptista Moura Santos, Prof. José Alberto Rodrigues Pereira Sardinha

Examination Committee
Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. José Alberto Rodrigues Pereira Sardinha
Members of the Committee: Prof. Maria Raquel Murias dos Santos Aires Barros

October 2018
Acknowledgments

To my parents, for all the love and support you’ve given me, in life in general, but especially throughout the past few months. I couldn’t have done it without you.

I would also like to acknowledge all the work and dedication of Pedro Rosado Dias, your contribution was crucial to the success of this work and I am very grateful.

Lastly, special thanks to Prof. Ana Moura Santos, Prof. Alberto Sardinha and Carla Boura Costa for their fantastic orientation in the past year.
Abstract

Mentoring programmes within universities have shown to improve students’ academic performance. The Mentoring Programme at Técnico Lisboa’s Taguspark campus has had a positive impact on students. Yet, it suffers from communication problems between the parties involved. A software solution named NAPP had been created to address these communication problems. NAPP’s Web application is a component that is meant to be used as a high-level performance analysis tool by the mentoring programme’s coordinator. The previous prototype of the Web application did not present the information necessary for the proper analysis of students’ academic performance. A new Web application was created which uses data visualisations to describe students’ academic performance. Further improvements were made to the other component of NAPP: the prototype of a mobile application to be used by mentors and mentees. The first beta test of the mobile application was conducted with a small group of students for 3 months, revealing that students used the application at least once a week, except for one week. A usability test on the new Web application showed high user satisfaction (average SUS score of 95). NAPP is ready to undergo a formal test which will allow to evaluate the system’s value for the Mentoring Programme in Taguspark.

Keywords: Mentoring Programme, Student Success, Student Support Systems, Data Visualisation
Resumo

Os programas de mentorado aplicados no contexto académico têm demonstrado melhorar o aproveitamento académico dos estudantes. O Programa de Mentorado no campus do Taguspark do Instituto Superior Técnico tem tido um impacto positivo nos estudantes. Ainda assim, existem problemas de comunicação entre as partes envolvidas. O NAPP é um sistema que foi criado anteriormente para abordar estes problemas. A aplicação Web do NAPP é uma componente para ser usada pela coordenadora do programa como uma ferramenta de análise do desempenho escolar dos alunos. O protótipo anterior da aplicação Web não mostrava a informação necessária para a correta análise do desempenho escolar. Assim, foi criada uma nova aplicação que usa visualizações de dados para descrever o desempenho escolar dos alunos. Foram também feitas melhorias ao outro componente do NAPP, o protótipo de uma aplicação móvel que irá ser usada pelos mentores e mentorandos. Foi realizado um primeiro teste beta à aplicação móvel, durante 3 meses, com um pequeno grupo de alunos, que revelou que estes usaram a aplicação móvel pelo menos uma vez por semana, com a exceção de uma semana. Um teste de usabilidade à nova aplicação Web mostrou elevada satisfação por parte dos utilizadores (uma média de 95 pontos na SUS). O NAPP vai agora ser submetido a um teste formal que permitirá avaliar a importância do sistema para o Programa de Mentorado no Taguspark.

Keywords: Programa de Mentorado, Sucesso Escolar, Apoio ao Estudante, Visualização de Dados
Contents

List of Tables xi
List of Figures xiii
Listings xv
Acronyms xvii

1 Introduction 1
   1.1 Key Contributions .................................................. 3
   1.2 Organization of the Thesis ........................................ 3

2 Background 5
   2.1 Related Work .......................................................... 5
      2.1.1 Chronus ......................................................... 5
      2.1.2 MentorCore ..................................................... 6
      2.1.3 Learning Analytics at Georgia State University ............. 7
   2.2 The NAPP Framework ................................................ 9

3 Core Work 11
   3.1 Approach ............................................................... 11
      3.1.1 Architectural Changes ......................................... 13
   3.2 Corrections and Improvements ....................................... 14
      3.2.1 Correcting the Data Model ..................................... 14
      3.2.2 Improving the Mobile Application ............................ 24
   3.3 The New Web Application ........................................... 30
      3.3.1 The Courses feature ........................................... 30
      3.3.2 The Students feature .......................................... 34
      3.3.3 The Alerts feature ............................................ 40
      3.3.4 The Dashboard feature ........................................ 43
      3.3.5 The Feedback feature ......................................... 44
      3.3.6 The REST API ................................................ 45
   3.4 Deployment ............................................................ 48

4 Evaluation 49
   4.1 Beta testing the mobile application ............................... 49
      4.1.1 Structured test ................................................. 49
      4.1.2 Unstructured test .............................................. 50
   4.2 Web application usability test ..................................... 51
List of Tables

3.1 Old data model – fields shared between mentors and mentees ..................... 14
3.2 Old data model – mentor-exclusive fields .............................................. 15
3.3 Old data model – mentee-exclusive fields .............................................. 15
3.4 Improved data model – fields of a document in the Feedback database ............ 16
3.5 Improved data model – fields of a document in the Tasks database ................ 18
3.6 Improved data model – fields of a document in the Courses database ............ 19
3.7 Improved data model – fields of a document in the Users database ............... 21
3.8 Improved data model – fields of a document in the Log database .................. 23
3.9 Improved data model – value of detail field depending on the value of the actionID field of a document in the Log database ........................................ 23
3.10 Fields of the coordinator’s document ...................................................... 41
3.11 The REST API endpoints ..................................................................... 47

4.1 The average SUS scores obtained by the Web application ......................... 54
List of Figures

2.1 Initial NAPP architecture ................................................................. 10
3.1 Current NAPP architecture ............................................................ 13
3.2 Old data model – a scenario where a replication conflict occurs and data are lost ................................................................. 17
3.3 Authentication in V1 – login screen (left) and sign-up screen (right) .......................................................................................... 25
3.4 Authentication in V2 – entry screen (left) and login screen (right) .............................................................................................. 26
3.5 Authentication in V2 – unknown user tries to sign up and is denied access (left); enrolled user tries to sign up and is allowed access (middle) and can fill in their personal information (right) .............................................................................................. 26
3.6 Authentication in V3 – entry screen (left) and FenixEdu login screen (right) ................................................................................... 27
3.7 Integration of grade reports with the course’s evaluation method ............................................................................................. 28
3.8 Web application – the list of courses ......................................................... 30
3.9 Web application – the page of the “Álgebra Linear” course, showing information about one of the course’s evaluations .............................................................................................. 32
3.10 Web application – adding a course requires filling in basic information (top) and then the evaluation method (bottom) .............................................................................................. 33
3.11 Web application – the students list .......................................................... 34
3.12 Web application – the Mentees tab of the mentor’s view ................................................................. 35
3.13 Web application – the Grades tab of the mentee’s view ................................................................. 36
3.14 Web application – the Study Tools tab of the mentee’s view ................................................................. 37
3.15 Web application – the Mentor tab of the mentee’s view (top) which allows to switch the student’s mentor (bottom) .............................................................................................. 38
3.16 Web application – the page to enrol mentors in the MP-TP ................................................................. 39
3.17 Web application – the page to view unregistered mentors and mentees ................................................................. 39
3.18 Web application – the Alerts panel ............................................................. 43
3.19 Web application – the Dashboard ............................................................... 44
3.20 Web application – the Feedback page ........................................................ 45
4.1 The activity logs generated during the unstructured test ................................................................. 51
4.2 The ten items on the System Usability Scale ......................................................... 54
A.1 A BPMN model of the MP-TP’s process for academic success prior to the implementation of NAPP .............................................................................................. 62
B.1 A BPMN model of the MP-TP’s process for academic success ................................................................. 64
3.1 An example of a user's grades field in the old data model, in JSON notation .......................... 18
3.2 The structure of each evaluation in the course's evaluationMethod field, in TypeScript notation for declaring object interfaces ................................................................. 19
3.3 The structure of each grade in the user's grades field in the improved data model, in TypeScript notation for declaring object interfaces ................................................................. 19
3.4 The structure of the roles field, in TypeScript notation for declaring object interfaces .............. 20
3.5 An example of a user's roles field, in JSON notation ................................................................. 20
3.6 The structure of the AlertsPreference entity, in Typescript notation for declaring object interfaces ........................................................................................................................................ 40
3.7 The AlertsPreference that will trigger an alert when a student reports an negative grade, in JSON notation ........................................................................................................................................ 40
3.8 The structure of the Alert entity, in Typescript notation for declaring object interfaces .......... 41
Acronyms

FCM  Firebase Cloud Messaging. 13, 41, 48

GSU  Georgia State University. 5, 7–9

LMS  Learning Management System. 5

MP-TP  Mentoring Programme in Taguspark. 1–3, 5–7, 9, 11, 12, 19, 20, 22, 25, 30, 31, 34, 35, 37, 39–45, 50, 51, 55, 57, 62, 64

NAPE  Núcleo de Apoio ao Estudante. 1

PBKDF2  Password-Based Key Derivation Function 2. 24

SUS  System Usability Scale. 53, 54
Chapter 1

Introduction

Núcleo de Apoio ao Estudante (NAPE)\(^1\) was founded on 3rd December, 1990 under the name Gabinete de Apoio ao Estudante. NAPE was created through the combined efforts of Instituto Superior Técnico\(^2\) (also referred to as IST or Técnico Lisboa) and Direcção-Geral do Ensino Superior\(^3\), with the intent of improving Técnico Lisboa’s academic success rates. This initiative was inserted in Programa Nacional de Combate ao Insucesso Escolar no Ensino Superior [1], a nation-wide programme to reduce school failure in higher education promoted by the then Portuguese Ministry of Education.

From the beginning, NAPE had a clearly defined mission which was to promote the integration of students at Técnico Lisboa. Presently, this mission is supported by actions such as: promoting the relationship between the students and the school; fostering initiatives that contribute to a better integration of students; supporting students’ initiatives; organising welcoming activities focused on first-year and international students; and publicising Técnico Lisboa’s offer in graduate programmes among high school students.

The main project within NAPE is the Mentoring Programme, a Técnico Lisboa pioneer project first implemented in 1996. The Mentoring Programme allows to bring students from different academic years of the same academic degree closer together. It focuses on the welcoming, integration, and assistance of students admitted for the first time to Técnico Lisboa, who are mainly first-year and international students. With the aid of NAPE’s mentors and guides – who are typically volunteer, second- or third-year students – the new students receive personalised assistance during their first contact with Técnico Lisboa, as well as through the remainder of the academic year.

The Mentoring Programme operates in the two \textit{campi} of Técnico Lisboa where students attend classes: the Alameda campus and the Taguspark campus. In the academic year 2011/2012 it went through a revamp in the Taguspark campus. The Mentoring Programme in Taguspark (MP-TP) uses a different strategy for accepting mentors: they are still volunteer, but they are interviewed beforehand in order to be accepted into the MP-TP. Another differentiating factor of the MP-TP is that the academic performance of new students is closely followed by a counsellor, who is the coordinator of the mentoring programme. Since the restructuring of the Mentoring Programme in Taguspark, the programme has had a positive impact on the academic success of first-year students at the campus.[2]

In order to achieve the intended goals of academic success and integration, the MP-TP depends on the exchange of information between three parties: the MP-TP’s coordinator, mentors, and mentees. Despite the positive impact of the programme on first-year students, there are admitted communication problems between the three parties involved.

\(^1\) http://nape.tecnico.ulisboa.pt/en/nape
\(^2\) https://tecnico.ulisboa.pt/en/
\(^3\) http://www.dges.gov.pt/
One of the most important flows of information in the MP-TP is the reporting of grades by mentees. The monitoring of mentees’ grades and the help it prompts are key aspects behind the success of the programme. Yet, the main communication problem that was described concerned this flow.

The monitoring of mentees’ grades is carried out by the MP-TP’s coordinator. The coordinator produces and analyses a spreadsheet with the grades reported by the mentees. By analysing the negative grades reported, the coordinator determines if a mentee is at risk of academic failure. Depending on the mentee’s situation, the coordinator decides whether to ask the respective mentor to help the mentee, or to interview the mentee and provide personalised support (or neither in case the mentee did not report any negative grade).

For the coordinator to receive the grades, the reporting of grades flows in a sort of “boomerang” fashion. The MP-TP’s coordinator has to message the mentors for their mentees’ grades, who will then message their mentees asking for the grades. The mentees then have to report their grades to their respective mentors, and afterwards the mentors forward their mentees’ grades back to the coordinator.

This process, which is illustrated in detail in Figure A.1, is where the main communication problem lies. The problem is divided into two parts: the reporting of grades is slow due to the “boomerang” flow, and the analysis of grades is also slow due to delay associated with using a very basic spreadsheet. This compromises the ability to provide help to at-risk mentees in time, so that they can get back on track before the next evaluations.

In this process, the reporting of grades relies almost completely on the exchange of emails between the three parties. In the academic year 2017/2018, the last portion of the reporting flow (from mentors to the coordinator) was modified: instead of emailing their mentees’ grades to the coordinator, the mentors write them directly on a collaborative spreadsheet in Google Sheets. This is then the spreadsheet that the coordinator uses to analyse the grades. This change allowed for a slight reduction of the delay: the spreadsheet is now produced in a distributed manner; but the time associated with the “boomerang” flow did not decrease, nor did the time of analysing the spreadsheet.

The NAPP framework is a software solution that was developed as the Master’s degree thesis of Pedro Veiga, to try to tackle the problems of the Mentoring Programme in Taguspark.[3] Its two main components are a mobile application to be used by mentors and mentees, and a Web application to be used by the MP-TP’s coordinator. A prototype of the mobile application was made and tested for usability with a small group of students. It supports the reporting of grades by mentees, and the monitoring of mentees’ performance by mentors. It allows students to submit feedback about the MP-TP, and also includes study guidance tools to assist students in their academic life.

With NAPP’s original version, mentees report their grades using the mobile application, which makes them instantly available to the mentors and MP-TP’s coordinator. This will eliminate the “boomerang” flow associated with the reporting of grades, and is expected to greatly improve the communication process in reporting grades. [4] Figure B.1 shows the changes to the MP-TP’s process for academic success once NAPP is used in the programme.

A prototype of the Web application was also made, but it did not undergo any kind of testing. NAPP’s Web application is intended to serve as a performance analysis tool for the MP-TP’s coordinator. Yet, the first prototype displayed only an overall view of the data in a table, and a small amount of aggregated data – such as the total amount of negative and positive grades reported. This does not satisfy the coordinator’s needs for analysing each student’s academic performance, and it is even a downgrade in terms of experience and efficiency because the table cannot be manipulated like a spreadsheet. Moreover, the suggestions submitted by mentees in the mobile application cannot be viewed in the prototype of the Web application.

The first version of NAPP had other problems besides the prototype of the Web application. NAPP was not integrated with the evaluation methods of Técnico Lisboa’s courses: the grades reported by
mentees were associated only with a course, and not with a specific evaluation of the course (e.g. tests, projects, exam, etc.) This association is critical to the correct analysis of the mentees’ academic performance by the coordinator.

1.1 Key Contributions

This work presents the various enhancements made to NAPP’s original version. The data model was redesigned for improved scalability. The mobile application was subject to significant modifications, amongst them the integration with the evaluation methods of Técnico Lisboa when reporting grades, and the overhaul of the authentication process which now uses Técnico Lisboa’s Central Authentication Service. The inadequate first Web application was discarded and a new one was created according to the requirements of the MP-TP’s coordinator. The new Web application underwent a usability test. Moreover, two beta tests were conducted on the mobile application. The first test was carried out over a short period of time, and it included a very small number of students. This test allowed to diagnose several problems in the existing prototype. These problems were corrected and a second test was conducted. This last test had the duration of three months and included a slightly higher number of students.

1.2 Organization of the Thesis

This thesis is divided into five chapters. Chapter 2 introduces the related work, including two mentoring software solutions and the case of a successful use of academic data for the benefit of students. It also introduces NAPP, as it was originally developed. Chapter 3 explains all the work that was done to enhance NAPP and Chapter 4 presents the various phases of evaluation. Lastly, Chapter 5 presents the conclusions of this work and outlines the future work.
Chapter 2

Background

2.1 Related Work

Organisations around the world are progressively choosing to use specialised mentoring software to enrich the experience of participants taking part in mentoring programmes. These systems focus mostly on enhancing communication between mentors and mentees, along with providing programme administrators with management and monitoring capabilities to assess the progress and results of the mentoring programme.

These mentoring software solutions concentrate on the interactions between mentors and mentees, and, even when they offer variants that target mentoring programmes in higher education, they do not include any capabilities for directly analysing students' academic progress. Since this is one of the fundamental aspects of the MP-TP, there was a need to investigate another topic which relates deeper with students' academic success: learning analytics.

Learning analytics is the use of data, analysis, and predictive modelling to improve teaching and learning.[5] In recent years, higher education organisations have started to take advantage of the large amounts of data generated by their Learning Management Systems (LMS) to provide insights on how to improve learning.[6] Georgia State University is a recognised successful case of the use of learning analytics to improve student success, which has been featured in a 2016 report from the White House.[7]

In the following sections, two prominent software solutions for mentoring programmes are presented, as well as the successful case of Georgia State University with learning analytics.

2.1.1 Chronus

Chronus\(^1\) is a mentoring software used by many organisations across the world. Although Chronus was initially developed for mentoring programmes in the context of professional organisations, presently it can also be applied to mentoring programmes within universities. The solutions offered are customisable to different types of mentoring programmes in education, such as, Peer Mentoring, Alumni Mentoring, and Career Mentoring.[8]

In Peer Mentoring, students who are at the beginning of their graduate programmes are paired with other students of the same department. The objective of peer mentoring is for students to receive academic support and guidance from their peers, typically during their first year at university. The Mentoring Programme in Taguspark fits into this category of mentoring. In Alumni Mentoring, students are paired with alumni of the school; and in Career Mentoring they are paired with professionals in their field of interest, who are not necessarily alumni of the school. In these two latter types, the goal is for students

\(^1\)https://chronus.com/
to receive academic and career advice from people who share a common area of interest with them, and it also poses an opportunity for developing and practising networking skills.

Chronus has been implemented within these different types of mentoring programmes in universities such as the MIT Sloan School of Management, and the Leeds School of Business of the University of Colorado, Boulder.

Chronus’ distinguishing feature is its MatchIQ® technology, which uses mentor and mentee information to suggest suitable matches. According to Chronus’ guidelines on how to start a high-impact mentoring programme, when dealing with large numbers of mentors and mentees, as is usual in universities, the best option is to use intelligent matching because it expedites the matching process.[9] Effective matching is vital in Career Mentoring and Alumni Mentoring, because the ultimate goal of these programmes is for mentees to integrate the job market. Effective matching can also be relevant in Peer Mentoring, however, this technology isn’t particularly useful in the context of the Mentoring Programme in Taguspark. The Taguspark campus of Técnico Lisboa enrols no more than 250 new undergraduate students at the start of each academic year, and the number of mentors in the programme is usually around 80, making the numbers too small to require intelligent matching. Thus, the matching has been randomised since the beginning of the programme, with the only constraint being that the mentor and the mentee must belong to the same academic degree. It is assumed that this randomisation has had no meaningful impacts on the results of the programme.

Within Chronus, mentors and mentees can engage in the programme through a Web application and native mobile applications for smartphones and tablets. These applications offer not only direct communication tools between mentors and mentees, but also mentoring resources for new mentors, as well as a forum where participants can submit questions and answers.

In what concerns monitoring and management features, Chronus provides workflow management, which enables mentoring programme administrators to create workflows for mentors and mentees, defining tasks and milestones to better guide them through the programme. Moreover, mentoring programme administrators also have access to tools which focus on helping to measure programme impact. One of these tools is a dashboard which contains real-time status reports regarding the mentoring programme, including information such as:

- A summary of the number of mentors and mentees in the programme.
- A mentor engagement report, which contains a summary of the number of “mentoring connections,” for example, meetings held with each mentor.
- A Programme Health report, which consists of a detailed report that helps in assessing the progress made by the participants of the programme.
- Suggestions on how to improve the mentoring programme.

Another tool offered to mentoring programme administrators is an integrated survey tool. At the end of the programme, mentors and mentees answer a short survey which helps administrators assess the programme’s level of success, engagement, and retention.

### 2.1.2 MentorCore

Launched in 2013, MentorCore is a mentoring software developed by CiviCore in collaboration with MENTOR: The National Mentoring Partnership.[10] The software is available in two flavours, suitable for two different types of mentoring: MentorCore Youth, for youth-based programmes; and MentorCore Higher Ed, for university programmes.[11]
Using MentorCore Higher Ed, mentoring programme administrators can create custom mentor and mentee application forms, as well as approve or deny applicants from entering the mentoring programme.

The software provides matching capabilities with customisable matching criteria, to be defined by administrators. MentorCore Higher Ed supports matching options which include peer matching, and student to alumni, professionals, or faculty; and thus, it can be used for Peer Mentoring, Alumni Mentoring, and Career Mentoring programmes. As said before, intelligent matching is not relevant in the context of the Mentoring Programme in Taguspark.

Unlike Chronus and NAPP, MentorCore does not provide a mobile application to programme participants. Instead, a Web application allows mentors and mentees to apply to the mentoring programme, find suitable matches, and engage in the programme.

This mentoring software offers some monitoring and management features to be used by mentoring programme administrators, including:

- Simple visualisations of programme data, which allow to analyse programme outcomes and objectives,
- Creation of customisable reports, which can be exported into Microsoft Excel,
- Creation of surveys to be answered by mentors and mentees.

Moreover, it allows mentoring programme administrators to upload resources and documents for mentors. It does not, however, allow administrators to schedule tasks to be performed by mentors and mentees.

Like these two systems, NAPP is intended to be a technological solution that promotes the communication between all the parties involved in a mentoring programme. However, what sets NAPP apart from Chronus and MentorCore is the special attention to the students’ academic success. None of these other systems provide modules that have the explicit focus set on following the academic track record of participants – not even Chronus, which has been applied in academic contexts several times before.

### 2.1.3 Learning Analytics at Georgia State University

In the years leading up to 2012, academic advisement at Georgia State University (GSU) was showing signs of decay. The university supported six advising offices which had developed independently across multiple colleges, and thus had little coordination, no sharing of records, and no common training. This separation, along with a 700 to 1 student-to-adviser ratio, resulted in an inefficient and ineffective academic advisement system which often failed to provide assistance to GSU’s large number of at-risk students (66% non-white, 33% first-generation students, and 53% receiving Pell Grants). In order to be able to more effectively support its students, GSU decided to modernise its advisement structure.

The university started by hiring new advisers, bringing the student-to-adviser ratio to 300 to 1, the national standard in the United States. A new, campus-wide, central advising office was also built, in which advisers representing every college and every major provide academic advisement to students.

Additionally, a Web-based advising platform was built in collaboration with the Education Advisory Board (EAB). The platform consists of an early-warning system, named Graduation and Progression.
Success (GPS, or GPS Advising), which tracks GSU’s 30,000 students on a daily basis, and applies ten years of historical data (2.5 million student grades) to create alerts and predictive analytics.[13] These alerts and analytics are delivered to adviser dashboards every morning, prompting student meetings with advisers, as well as providing guidance during those meetings. The platform keeps track of meetings with advisers and allows them to write notes on a permanent record available to all advisers in the university, making advisement records follow students through major and college changes.

Furthermore, the GPS Advising system enables advisers to rapidly create custom reports based on a large number of variables, for instance, the number of alerts, a specific GPA score, students’ academic year (first year, second year, etc.) The system also tracks general university analytics, which are filtered by university, college, and department. These analytics serve as a measure to assess the impact of the interventions prompted by the GPS system, and may also encourage further interventions, for instance, when a significant number of students in a certain major is off path for graduation, the department might invite those students to group tutoring and advisement sessions.

Georgia State University and EAB began developing the GPS Advising system in January 2012. The system went live in August 2012, and EAB provided common training to the university’s advisers. The system is hosted outside the university’s facilities.[12]

In 2015, GSU began working on a mobile application that would make some of the information generated by the GPS Advising platform available to students.[14]

In a 2017 status report, Georgia State University states that the GPS Advising system allowed to greatly improve the university’s graduation rate4. [15] In 2003, GSU reported a university-wide graduation rate of 32%. Yet, the rates were lower for at-risk students: 22% for Latinos, 29% for African Americans, 18% for African American males, and with students who received Pell Grants graduating at much lower rates than their non-Pell peers. By 2017, the university-wide graduation rate has increased by 22 percentage points, and now stands at 54%. Achievement gaps were eliminated for at-risk students, whose graduation rates are now: 55% for Latinos, 58% for African Americans, and 54% for students receiving Pell Grants who make up 58% of the university’s undergraduate student population. Additionally, GSU reported that time-to-degree has been reduced by one semester since the implementation of GPS Advising, which means that students spend one less semester in college to graduate.

Alerts

The GPS Advising system uses data mining techniques to generate alerts which serve as indicators that the student has gone off path for graduation. The system contains over 800 alerts derived from ten years of historical data.[13] These alerts represent risk behaviours of students, such as:

- Signing up for a course that is not part of the student's declared major,
- Failing to obtain a minimum grade in a certain course which is critical to success in their major, or
- Not having completed a course at a particular point in time in the academic year.[12]

It is important to note that in the specific case of failure to earn a minimum grade in a critical course, a minimum grade is regarded differently from a passing grade. A passing grade is simply the grade required to complete the course. A minimum grade is the grade that should be earned in a critical course to ensure that the student will be successful in that major. For example, the data show that to graduate in nursing, students need to earn a grade of at least B+ in the first mathematics course,

---

4The graduation rate is the percentage of students enrolled in the university who completed a four-year college degree.
5In the United States, students who enrol in an undergraduate degree are usually not required to immediately choose their major, and may freely sign up for any course, in any desired order. Students are generally required to declare a major by the end of their second academic year, if they have not done so already.
while the passing grade is C. This minimum grade is computed by the GPS system for every course by analysing past students’ grades, and an alert is delivered to an adviser’s dashboard when a student fails to achieve the minimum grade.

Georgia State University is a very successful case that highlights how data can be used to benefit students. The GPS Advising system shares some aspects with NAPP, such as the idea of delivering relevant alerts to the adviser’s dashboard, or the visualisation of each student’s progress in that same dashboard. Yet, the success of the GPS Advising initiative depends on the students’ willingness to actively interact with their assigned adviser. When students enter the higher education system, they can often feel overwhelmed and lost. Peer Mentoring in higher education helps new students feel welcome and integrated, which often translates into their academic success. The core goal of NAPP is to join technology, Peer Mentoring and academic data to ultimately contribute to students’ academic success.

2.2 The NAPP Framework

The NAPP framework was developed as the master’s degree project of Pedro Veiga in 2017. From an architectural perspective, the NAPP framework is designed from the beginning to consist of three elements: the mobile application, the Web application, and a server which hosts the database that keeps all the data generated by the applications.

The mobile application was built using the Ionic Framework, which is an open-source Software Development Kit for hybrid mobile development. The Ionic Framework is built on top of Angular and Apache Cordova. It provides tools to develop cross-platform mobile applications using Web technologies like CSS, HTML5, and JavaScript. Mobile development with the Ionic Framework has the benefits of maintaining a single codebase for all mobile platforms, and lower development time when compared with the time necessary to develop native applications. Applications developed using the Ionic Framework are distributed through native distribution platforms (“app stores”), such as Google Play (Android) and App Store (iOS).

The mobile application is divided into two views: the mentee view and the mentor view. It allows mentees to report their grades – though this feature was not integrated with the evaluation methods of Técnico Lisboa’s courses – and mentors to follow their mentees’ academic evolution. It also includes study guidance tools for mentees, such as a list of tasks integrated with a Pomodoro timer. Mentors can also add tasks to their mentees’ task lists through their view in the mobile application. The mobile application also has a feedback feature which mentees can use to submit their suggestions for improving the MP-TP.

The Web application uses a front-end dashboard template, ng2-admin, which is built on top of Angular, and provides tools to create charts and tables. The Web application consists of two pages. The first page displays a small amount of aggregated data – such as the total amount of mentees and mentors in the programme, or the total amount of positive or negative grades reported. The second

---

6https://ionicframework.com/
7https://angular.io/
8https://cordova.apache.org/
9The Pomodoro Technique is a time-management method created by Francesco Cirillo. The technique uses a timer to divide work into 25-minute intervals, called “Pomodoros,” followed by short breaks. After four “Pomodoros” have been completed, the person should take a 20 to 30 minute break – this makes a full Pomodoro cycle.[16]
10https://akveo.github.io/ng2-admin/
page contains a table that displays the student information, each row corresponding to a student. This table allows the coordinator to manipulate student data: sort information by attributes and edit incorrect information.

The server hosts an Apache CouchDB\(^\text{11}\) database. CouchDB is open-source database software. It has a document-oriented NoSQL database architecture which focuses on scalability, reliability, and ease of use. The CouchDB database hosted on the server stores all the data related with the NAPP framework. Local data storage on the mobile application and on the Web application is done through PouchDB\(^\text{12}\). PouchDB is open-source JavaScript database software which was created to facilitate the development of offline applications. It also has a document-oriented NoSQL database architecture. PouchDB stores data locally on the device, and then syncs those data to the server-side CouchDB database when an Internet connection is available. This automatic synchronisation of data follows the Couch Replication Protocol, which is a protocol for synchronising data between two peers over HTTP. Additionally, PouchDB provides a convenient API for querying the database.

The system architecture, as described above, is presented in Figure 2.1.

---

\(^{11}\)http://couchdb.apache.org/

\(^{12}\)https://pouchdb.com/
Chapter 3

Core Work

3.1 Approach

From the beginning, it was known that the evaluation of this work was highly dependent on whether there would be real data to analyse how valuable NAPP will be to the Mentoring Programme in Taguspark. For this reason, the first objective was to get the mobile application to a state where it was ready to be used by mentors and mentees. Since the mobile application was initially developed as a prototype, it had only been tested for usability; and NAPP as a whole had only been run on a local computer environment.

At that time, a considerable flaw of NAPP was related to the reporting of grades: the grades reported by students were not linked to an evaluation, which would make it very difficult for the MP-TP’s coordinator to follow the students’ progress accurately. The approach chosen was to model the courses’ information, including the evaluation methods, and store it in a database. NAPP’s mobile application was then modified to allow students to choose an evaluation when reporting a grade, not only the course.

With these changes, NAPP was ready for a first round of beta testing. The system was tested with a small group of students for three days. The database was set up on a desktop computer and the mobile application was built and installed on the students’ mobile devices. The students were given a small guide which had a set of tasks to complete on given days, along with the expected results. The goal was to make sure that the system behaved as expected. Because of the very methodical nature of this test, it was named the “structured test”.

During the structured test some critical problems were found that needed to be fixed. The most severe of these problems had to do with NAPP’s data model. At that point, all the data generated by users were stored in a single database. Section 3.2 will explain in detail why this is problematic. Briefly, this approach of centralising all the user-related data in a single collection of documents does not scale well: firstly, it makes queries to the database unnecessarily complex and slow; secondly, it is very prone to creating replication conflicts. A replication conflict happens when CouchDB is trying to sync a remote database with the local databases on the users’ devices, but the data being synced are incompatible (this happens because of concurrent writes), so one of the versions is discarded. Other problems were found during the structured test, for example, one problem had to do with the interaction flow of the authentication process in the mobile application. The discovery of these problems prompted an extensive redesign of NAPP’s data model and further improvements to the mobile application, which are covered in Section 3.2.

With these problems fixed, the next objective was to test the mobile application in an environment closer to the reality of the MP-TP. During three months of the second semester of the academic year 2017/2018, the mobile application was used by a small group of students. The students were presented
the features of the mobile application but were not given any guide to follow, they were free to use the mobile application as they pleased. Because of the more informal nature of this test, it was named the “unstructured test”. The initial objective was to be able to make this test with a significant number of students, but because of limitations that will be explained in the Evaluation chapter, it was only possible to gather a smaller group of students.

While the tests took place, another core part of this work was being developed. NAPP’s Web application is a fundamental component of NAPP, because it is the element that aims to help the MP-TP’s coordinator to analyse the students’ grades more efficiently. The problem was that the existing Web application did not suit the needs of the coordinator. Although the user information was displayed in a table, this table did not resemble the spreadsheet already used by the coordinator and could not be manipulated like that spreadsheet. The previous Web application was developed using the front-end dashboard template ng2-admin, and this template was sufficient for the very few features of the application. However, due to the template’s high complexity and poor documentation, it was not suitable for the continued improvement that NAPP requires. For these reasons, the existing Web application was discarded and a new one was built according to the coordinator’s requirements.

The new Web application was developed with Angular, without the use of any template. The Web application employs charts to facilitate the visualisation of information. The choice of charts was guided by best practices in information visualisation.[17] The Chart.js1 library is used to draw the charts. Chart.js is a charting library for JavaScript that has support for various types of charts and interactivity. The Web application is divided into five features:

- The Courses feature allows to create and manage courses, and to explore and analyse the results of evaluations,
- The Students feature helps to follow students’ progress, and it also has some functionality to manage the mentoring programme,
- The Alerts feature delivers alerts about relevant events within the mentoring programme,
- The Dashboard feature provides an at-a-glance view of the mentoring programme,
- The Feedback feature allows to view the feedback submitted by students.

Besides these components, a REST API was developed to fulfil some requests that would put a heavy load on the client machine if they were computed by the Web application. The Web application was tested for usability with small groups of people.

After the end of the structured and unstructured tests, another important improvement was made to the mobile application. The authentication process became integrated with the authentication service of Técnico Lisboa, to allow the students to use NAPP with their academic credentials.

All these tests and enhancements allowed to get NAPP into a state where it is ready to undergo a bigger and more formal test, in a real environment. The goal of this test is to evaluate whether NAPP contributes positively to the MP-TP by evaluating some key metrics. The NAPP system will be tested with two groups of students: the experimental group will use NAPP, and the control group will follow the traditional method of the MP-TP. This test has been planned and will take place during most of the first semester of the academic year 2018/2019.

1http://www.chartjs.org/
3.1.1 Architectural Changes

The approach explained in the previous section prompted some modifications in the system architecture. The server no longer runs only CouchDB.

The REST API also runs on the server, and it answers requests from the Web and mobile applications. The REST API was developed in Node.js, which is a run-time environment that executes JavaScript code outside of the browser. Because the REST API runs in Node.js, it can use PouchDB to communicate with CouchDB for database access and sync.

The Alerts feature of the Web application requires that a worker runs on the server. This worker was developed in Node.js and it also uses PouchDB to connect to CouchDB for database access and sync. The worker processes the records of relevant actions and generates alerts. The alerts generated are written to the database, but there is also the possibility to receive them through push notifications. For this purpose, the Alerts worker uses Firebase Cloud Messaging (FCM), which is a cross-platform cloud solution for messaging provided by Firebase, a subsidiary of Google.

The Web application has database access through PouchDB – it keeps local replicas of most databases, which sync with the remote server databases. As for the mobile application, each client device running the application will have its local replicas of the databases, and PouchDB will take care of keeping them in sync with the remote server databases.

Figure 3.1 illustrates the current NAPP architecture, as described here.
3.2 Corrections and Improvements

3.2.1 Correcting the Data Model

CouchDB is a database management system for document-oriented NoSQL databases. Each database is a collection of documents stored as JSON objects. JSON is a text format for the serialization of structured data. JSON objects consist of key-value pairs, which will be referred to as “fields”. The field’s key must be a string, and the field’s value must be a valid JSON data type (string, number, object, array, boolean or null). A CouchDB document always has two mandatory fields: _id and _rev. The _id is the field that identifies the document in the database, so the value must be unique within the database where the document is stored. The _rev field is used for replication, a central feature of CouchDB. “Rev” stands for revision, and the _rev field serves as a version tag.

As was described in Section 3.1.1, each client running the mobile application will have its own replicas of the databases. PouchDB ensures that each mobile application will keep its own replicas in sync with the server-side databases. However, when more than one client is trying to sync with a server-side database, conflicts can happen (this is explained in much more detail throughout the following sections). CouchDB and PouchDB do not provide automatic conflict resolution – instead, the task of resolving conflicts is left to the developer. This task can be quite difficult, as it requires comparing the two conflicting versions and deciding which one to keep, or, if necessary, merging the two versions (which is even more difficult if the document has a large number of fields). A better approach resulted from modifying the data model to simply avoid conflicts.

**The old data model**

In the old data model, all the data were stored in a single database called “napp_data”. There was a document in the database for each user. A user could be of one of two types: mentor and mentee. The studentType field was used to distinguish between the two types. Some fields were common to both types of users (Table 3.1), others were exclusive to mentors (Table 3.2) and to mentees (Table 3.3).

<table>
<thead>
<tr>
<th>Key</th>
<th>Value Type</th>
<th>Description</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>string</td>
<td>The user’s unique identifier</td>
<td>Value is the student’s IST ID</td>
</tr>
<tr>
<td>studentType</td>
<td>string</td>
<td>The type of student</td>
<td>Possible values: &quot;mentor&quot; or &quot;mentee&quot;</td>
</tr>
<tr>
<td>istID</td>
<td>string</td>
<td>The student’s IST ID</td>
<td></td>
</tr>
<tr>
<td>firstName</td>
<td>string</td>
<td>The student’s first name</td>
<td></td>
</tr>
<tr>
<td>lastName</td>
<td>string</td>
<td>The student’s surname</td>
<td></td>
</tr>
<tr>
<td>phoneNumb</td>
<td>string</td>
<td>The student’s phone number</td>
<td></td>
</tr>
<tr>
<td>locality</td>
<td>string</td>
<td>The student’s locality of residence</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>string</td>
<td>The student’s email address</td>
<td></td>
</tr>
<tr>
<td>course</td>
<td>string</td>
<td>The acronym of the student’s degree</td>
<td>Should have been named &quot;degree&quot; instead</td>
</tr>
<tr>
<td>password</td>
<td>Object or number []</td>
<td>The encrypted password</td>
<td>Value differed if the password was generated on an Android or an iOS device</td>
</tr>
<tr>
<td>salt</td>
<td>string</td>
<td>The cryptographic salt</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Old data model – fields shared between mentors and mentees

²The _rev field is omitted from all data model tables for simplicity.
Table 3.2: Old data model – mentor-exclusive fields

<table>
<thead>
<tr>
<th>Key</th>
<th>Value Type</th>
<th>Description</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mentees</td>
<td>string[]</td>
<td>The student's mentees</td>
<td>Contains the _ids of the mentees</td>
</tr>
<tr>
<td>menteesFeed</td>
<td>string[]</td>
<td>A feed of events completed by mentees</td>
<td>Contains sentences describing the events</td>
</tr>
<tr>
<td>numAssignedTasks</td>
<td>number</td>
<td>The number of tasks that the mentor assigned to mentees</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 3.3: Old data model – mentee-exclusive fields

<table>
<thead>
<tr>
<th>Key</th>
<th>Value Type</th>
<th>Description</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mentor</td>
<td>string</td>
<td>The student's mentor</td>
<td>Value is the _id of the mentor</td>
</tr>
<tr>
<td>todoTasks</td>
<td>string[]</td>
<td>The tasks that the student has not completed</td>
<td>Each string contains only a description of the task. When the task was assigned by the mentor, the description is prefixed with “[Mentor]”</td>
</tr>
<tr>
<td>doneTasks</td>
<td>string[]</td>
<td>The tasks that the student has completed</td>
<td>When a task is completed, the task that was present in the todoTasks array is simply moved to this array</td>
</tr>
<tr>
<td>grades</td>
<td>Object[]</td>
<td>The grades that the student has reported</td>
<td>Each object in the array has only the fields “grade” (number, the grade) and “course” (string, the name of the course)</td>
</tr>
<tr>
<td>positive</td>
<td>number</td>
<td>The number of positive grades</td>
<td>–</td>
</tr>
<tr>
<td>negative</td>
<td>number</td>
<td>The number of negative grades</td>
<td>–</td>
</tr>
<tr>
<td>mean</td>
<td>number</td>
<td>The arithmetic mean of the student’s grades</td>
<td>–</td>
</tr>
<tr>
<td>feedback</td>
<td>string[]</td>
<td>The suggestions submitted by the mentee</td>
<td>Only the feedback text, no other metadata</td>
</tr>
<tr>
<td>checkmarks</td>
<td>number[]</td>
<td>The status of the Pomodoro timer in the User Interface</td>
<td>–</td>
</tr>
<tr>
<td>numCompletedPomodoros</td>
<td>number</td>
<td>The number of completed Pomodoro cycles</td>
<td>–</td>
</tr>
<tr>
<td>numDoneTasks</td>
<td>number</td>
<td>The number of completed tasks</td>
<td>–</td>
</tr>
</tbody>
</table>

This model had some problems, especially in what concerned the mentee-exclusive fields. The fact that every piece of information regarding or generated by a user was centralised in a single document made queries to the database unnecessarily complex when trying to combine data of all users. For example, in order to get all the feedback submitted by all the mentees in a single array of strings (basically merging every user’s feedback field into one array), one needed to create a database view associated with a “map-reduce” operation which, upon receiving a user’s document, would iterate through and emit every entry of the feedback array. A more thoughtful approach can be used, and one can simply store each new feedback entry as its own document of a separate Feedback database (which is what is done as per the improved data model). Then the same query is as simple as retrieving all the documents in the Feedback database.

As was briefly mentioned before, this model was very prone to creating replication conflicts. This is
due to the fact that a user’s document will be accessed by more than one client mobile application. In fact, if the user is a mentee, their client application will have access to the user’s document as well as the document of the user’s mentor. Similarly, if the user is a mentor, their client application will have access to the user’s document as well as the documents of the user’s mentees. As mentioned in Section 3.1.1, PouchDB syncs local replicas with the remote CouchDB databases when the client device is online. If a document is modified by more than one client while one (or both) of them is offline, then surely there will be a conflict when PouchDB tries to sync with the remote database once back online, and the conflicting version will be discarded. This happened, for example, with the field `toDoTasks`, because of the ability for mentors to assign tasks to their mentees. This problem was found due to the structured test, because there was a particular chore in the guide that called for users to concurrently create tasks while offline, which resulted in the loss of some tasks. This specific replication problem was solved with the introduction of a Tasks database, as will be explained in further detail below.

Additionally, documents in the “napp_data” database stored unnecessary information which was only related to local application logic, and which did not make sense to persist in a remote database – for example, the `checkmarks` field, which refers to the status of the Pomodoro timer on the User Interface.

The improved data model

The Feedback database

The Feedback database stores the feedback submitted by students. Each document in the database represents a suggestion. The date of submission is stored alongside the feedback in the form of a timestamp, which allows to sort the feedback by date – this was not possible in the old data model. Table 3.4 shows the fields of each document.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value Type</th>
<th>Description</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>_id</code></td>
<td>string</td>
<td>The feedback entry’s unique identifier</td>
<td>Follows the pattern <code>date-user</code>, the concatenation of both fields, to ensure default ordering by date while avoiding collisions</td>
</tr>
<tr>
<td>date</td>
<td>string</td>
<td>The entry’s submission date</td>
<td>Follows ISO 8601 standard for combined date and time in UTC</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>The <code>_id</code> of the user who submitted the feedback</td>
<td>–</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>The feedback text</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 3.4: Improved data model – fields of a document in the Feedback database

The Tasks database

As explained before, it was necessary to extract the tasks from the user document because of replication conflicts that were found in the aftermath of the structured test. Figure 3.2 shows the scenario that prompted the discovery of these replication conflicts. This scenario features the remote server database, two local replicas (mentee and mentor) on NAPP’s mobile application, and the mentee’s document. In this scenario, the initial state has the two user devices online, and thus the local replicas are in sync with the remote database – the three databases share revision “Rev 4” of the mentee’s document. In the old data model, all the newly created tasks were added to the `toDoTasks` field of the mentee’s document (for simplicity, this scenario omits all the other fields of the document, and it is assumed that they remain
unchanged). At one point, the mentee’s device goes offline and the mentee adds a task to `todoTasks` – this creates revision “Rev 5” of the mentee’s document. Since the mentee device is offline, this revision is not synced to the remote database. Meanwhile, the mentor assigns a task to the mentee, which is added to `todoTasks` – this creates revision “Rev 5′”, which is synced to the remote database because the mentor’s device is online. When the mentee’s device comes back online, the local mentee replica begins to sync with the remote database. The local revision “Rev 5” conflicts with the already synced revision “Rev 5′”. The remote database will reject “Rev 5” in favour of “Rev 5′” and the task created by the mentee will be lost, because there are no conflict resolution procedures in place.

To avoid the effort of creating and maintaining conflict resolution procedures, these replication conflicts were solved by modelling the “Task” entity. Instead of saving the tasks in the user document, each task is stored its own document in the Tasks database, and the `todoTasks` and `doneTasks` fields can be removed from the user document. The task document stores not only the text of the task but also other metadata. The text of the task is kept in the `description` field. A timestamp of the date of creation is kept in the `dateCreated` field. Instead of moving the task from the `todoTasks` field to the `doneTasks` field in the user document, the status of the task is saved in the `status` field; this field can have the values “pending” or “completed”. When the task is deleted by the user, the document is removed from the database. The `author` field stores the `_id` of the user who created the task, and the `target` field stores the `_id` of the user for whom the task is intended. To simplify filtering in the mobile application, an additional `type` field is stored, which allows to easily distinguish between a task created
Table 3.5: Improved data model – fields of a document in the Tasks database

by a mentee or a mentor. As an example, if the mentee creates a task in their own task list, the mentee will be the author and the target, and the type field will have the value “mentee”. If the mentor creates a task for a mentee, the mentor will be the author, the mentee will be the target, and the type field will have the value “mentor”. Table 3.5 presents the fields stored for each task, as described.

The Courses database

In order to be able to integrate the reporting of grades with the evaluation methods of Técnico Lisboa’s courses, it was necessary to store information about the courses themselves.

In the old data model, there were no data about the courses stored in any database. The mobile application simply hard-coded the names of first-semester courses – this meant that the information about the courses was not modifiable from outside the mobile application, but also that the application was not suited to be used in the second semester of the academic year. Additionally, by hard-coding only the names of the courses, it meant that the reported grades were not associated with an evaluation of the course – there was no notion of whether a grade reported corresponded to, for example, the first or second test. Listing 3.1 shows an example of what a user’s grades field looked like in the old data model.

```
1 "grades": [
2  {
3     "course": "Álgebra Linear",
4     "grade": 3
5  },
6  {
7     "course": "Fundamentos da Programação",
8     "grade": 15
9  }
10 ]
```

Listing 3.1: An example of a user’s grades field in the old data model, in JSON notation

These problems were solved by modelling the “Course” entity. Each course is a document in the Courses database. Besides the full name of the course, other course attributes are stored. Table 3.6 shows the fields stored for each course.
One of the most relevant fields is the `evaluationMethod`, which describes all the evaluations of the course. Listing 3.2 shows the structure of an evaluation. Each evaluation is an object which has a `name`, a `weight` that shows the contribution to the final grade of the course, and optionally a `date`. Another aspect that was important to model was the fact that not all evaluation grades fit the usual 0-20 scale, so it was also necessary to keep track of each evaluation's grade intervals (`minGrade` and `maxGrade`).

With the courses' information being stored in the Courses database, the reporting of grades can now be properly integrated with the evaluation methods. Listing 3.3 shows the new structure of each grade in the users' `grades` field, with data about the course and evaluation being stored alongside the grade.

Lastly, it should be mentioned that all the data regarding courses is to be added into the Courses database by MP-TP's coordinator through the Web application, and then presented by the mobile application in the screen where students report their grades.
The concept of user role

In the old data model, the role of a student in the MP-TP (whether the student was a mentor or a mentee) was represented by the studentType field. This approach had some limitations.

To begin with, there was nothing in the old data model connecting the studentType field to a given academic year — in the MP-TP, it is possible for a student to be a mentee in one year and a mentor in the next year. If the change of role was reflected in the user document by simply changing the value of the studentType field, one would lose the ability to track the history of the MP-TP. There might be a need to answer questions such as: How many mentees become mentors in the next year? How long do mentors stay in the MP-TP? With the old data model it was not possible to answer these questions.

Secondly, the mentor and mentees fields are directly connected to the user role – only mentees have mentors, and only mentors have mentees. As a consequence, these fields are also directly connected to the academic year. Thus, the data model should reflect this connection.

In the improved data model, user roles are defined on a per-academic-year basis. A user can only have one role per academic year. User roles correspond to being a mentor or a mentee. Listing 3.4 defines the structure of the roles field.

```typescript
interface Roles {
  [academicYear: string]: {
    role: "mentee";
    mentor: string;
  } | {
    role: "mentor";
    mentees: string[];
  }
}
```

Listing 3.4: The structure of the roles field, in TypeScript notation for declaring object interfaces

That is, the roles field is an object where each key corresponds to an academic year. For each academicYear key, there is an object that details the role for that year. If the role is "mentee", then the object will have another field called mentor (the _id of the mentor). If the role is "mentor", the object will have another field called mentees (an array with the _ids of the mentees). Listing 3.5 gives an example of a user's roles field. In this example, the user was a mentee in the academic year 2016/2017, and their mentor was the user whose _id is "ist145454". In the academic year 2017/2018, the user was a mentor, and their mentees were the users with _ids "ist199999" and "ist199998".

```json
"roles": {
  "2016/2017": {
    "role": "mentee",
    "mentor": "ist145454"
  },
  "2017/2018": {
    "role": "mentor",
    "mentees": [
      "ist199999",
      "ist199998"
    ]
  }
}
```

Listing 3.5: An example of a user's roles field, in JSON notation
<table>
<thead>
<tr>
<th>Key</th>
<th>Value Type</th>
<th>Description</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>string</td>
<td>The user’s unique identifier</td>
<td>Value is the student’s IST ID</td>
</tr>
<tr>
<td>istID</td>
<td>string</td>
<td>The student’s IST ID</td>
<td></td>
</tr>
<tr>
<td>firstName</td>
<td>string</td>
<td>The student’s first name</td>
<td></td>
</tr>
<tr>
<td>lastName</td>
<td>string</td>
<td>The student’s surname</td>
<td></td>
</tr>
<tr>
<td>phoneNumb</td>
<td>string</td>
<td>The student’s phone number</td>
<td></td>
</tr>
<tr>
<td>locality</td>
<td>string</td>
<td>The student’s locality of residence</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>string</td>
<td>The student’s email address</td>
<td></td>
</tr>
<tr>
<td>degree</td>
<td>string</td>
<td>The acronym of the student’s degree</td>
<td></td>
</tr>
<tr>
<td>studentNumber</td>
<td>string</td>
<td>The student’s IST number</td>
<td></td>
</tr>
<tr>
<td>roles</td>
<td>Roles</td>
<td>The user roles</td>
<td>As defined in Listing 3.4</td>
</tr>
<tr>
<td>grades</td>
<td>Grade [ ]</td>
<td>The student’s reported grades</td>
<td>As defined in Listing 3.3 Optional, this field exists only if the user ever had the role of mentee</td>
</tr>
<tr>
<td>menteesFeed</td>
<td>Object [ ]</td>
<td>A feed of events completed by mentees</td>
<td>Optional, this field exists only if the user ever had the role of mentor</td>
</tr>
</tbody>
</table>

Table 3.7: Improved data model – fields of a document in the Users database

The Users database

The Users database stores the documents that identify each user in the NAPP framework. It is similar to the main (and only) database in the old data model. Table 3.7 shows the fields of a user document.

As a consequence of the changes explained in the previous sections, the following fields were removed from the user document:

- studentType, mentor (previously a mentee-exclusive field), and mentees (previously a mentor-exclusive field) were integrated into the roles field
- toDoTasks and doneTasks were removed due to the introduction of the Tasks database
- feedback was removed due to the introduction of the Feedback database

Besides these, the following fields were also removed:

- positive, negative, mean (previously mentee-exclusive fields) were calculated every time the user submitted a new grade. However, they were being incorrectly calculated because they did not take into account the lower and higher limits of the evaluation grades (now minGrade and maxGrade). Because these values are only shown in the home screen of the mentee in the mobile application, it was decided that there was no need to store them in the database, and thus they are now calculated on demand by the mobile application
- numAssignedTasks (previously mentor-exclusive field), numCompletedPomodoros, numDoneTasks (previously mentee-exclusive fields) were not displayed anywhere on the mobile application and were only used by the old Web application (which was discarded). Currently there was no need to store these fields as they can be determined by querying the Log database (see next section), so they were removed. If in the future there is a need to keep these values, either for faster querying or to reduce querying costs, it is better practice to store them in a separate Aggregation database, not in the Users database
checkmarks (previously a mentee-exclusive field) was related to the visual status of the Pomodoro timer in the User Interface of the mobile application, and it should instead be stored locally on the user device.

password and salt were removed due to the integration with FenixEdu’s Central Authentication Service (see Section 3.2.2).

In addition, the course field was renamed to degree, because the previous name was an incorrect translation of the Portuguese word “curso”. The field studentNumber was also introduced because of a requirement from MP-TP’s coordinator, who requested to be able to search for students by their student number instead of their IST ID (the _id field). Like the _id field, studentNumber is also unique for each user.

The Log database

The Log database keeps a record of relevant actions carried out by users of NAPP. Each document in the database stores information about the action it represents. The Log database was created out of the necessity to visualise and, in the future, analyse the use of the mobile application by mentees and mentors.

A document in the Log database has the fields shown in Table 3.8. From the features provided by NAPP’s mobile application, five relevant actions are recorded, which are represented by the following values of the actionID field:

- "reportGrade" – when a mentee reports a grade
- "createTask" – when a mentee creates a task, or a mentor assigns a task to a mentee
- "completeTask" – when a mentee completes a task
- "feedback" – when a mentee submits feedback
- "pomodoroComplete" – when a mentee completes a Pomodoro (25-minute interval)

Then, depending on the actionID, each document also keeps a detail field, with extra information about the action carried out. Table 3.9 shows the structure of the detail field for each actionID value. Some actionID values will not be accompanied by a detail field.

In summary, the changes introduced to the data model allowed: to avoid database conflicts, and thus prevent data loss; to integrate the reporting of grades with the courses’ evaluation methods; to simplify database queries; and to visualise – and pave the way for future analysis of – the use of the mobile application by mentees and mentors.
<table>
<thead>
<tr>
<th>Key</th>
<th>Value Type</th>
<th>Description</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>string</td>
<td>The log record's unique identifier</td>
<td>Follows the pattern timestamp-user, the concatenation of both fields, to ensure default ordering by date while avoiding collisions</td>
</tr>
<tr>
<td>timestamp</td>
<td>string</td>
<td>The time of creation of the log record</td>
<td>Follows ISO 8601 standard for combined date and time in UTC</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>The _id of the user who generated the log record</td>
<td>–</td>
</tr>
<tr>
<td>actionID</td>
<td>string</td>
<td>An identifier of the action performed by the user</td>
<td>Possible values: “reportGrade”, “createTask”, “completeTask”, “feedback”, “pomodoroComplete”</td>
</tr>
<tr>
<td>detail</td>
<td>Object</td>
<td>Details about the action, which will depend on the actionID</td>
<td>This field only exists in documents whose actionID value has an entry in Table 3.9</td>
</tr>
</tbody>
</table>

Table 3.8: Improved data model – fields of a document in the Log database

<table>
<thead>
<tr>
<th>actionID value</th>
<th>detail structure (TypeScript notation for declaring interface properties)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“reportGrade”</td>
<td><code>detail: { courseID: string; evalName: string; grade: number; }</code></td>
</tr>
<tr>
<td>“createTask”</td>
<td>`detail: { taskType: &quot;mentor&quot;</td>
</tr>
<tr>
<td>“completeTask”</td>
<td>`detail: { taskType: &quot;mentor&quot;</td>
</tr>
</tbody>
</table>

Table 3.9: Improved data model – value of detail field depending on the value of the actionID field of a document in the Log database
3.2.2 Improving the Mobile Application

The following sections are dedicated to explaining the improvements that were made to NAPP’s mobile application. The mobile application received two considerable improvements. Firstly, the changes made to the authentication process, which got two new versions different from the initial state (V1): while the first new version (V2) was only an intermediate state, to fix the existing problems before the unstructured test (Section 4.1.2), the latest version (V3) is intended to be the definitive state of the authentication process. Secondly, the integration of the reporting of grades in the mobile application with the courses’ evaluation methods, using data from the Courses database created for this purpose. Other smaller enhancements were also made to the mobile application.

Authentication

Initial state (V1)

The first problem in V1 was the format in which the password was stored in the database. The encryption algorithm used was Password-Based Key Derivation Function 2 (PBKDF2). The native (Android and iOS) implementations of PBKDF2 have different output formats – either an object or an array – and the output of the algorithm was stored in the database without any further modifications. In cryptography, the standard way to verify if the login credentials are correct is to encrypt the input password and compare the result with the stored encrypted password. Because the output of the PBKDF2 algorithm differed on Android and iOS, this meant that if the user created the account on an Android device, they would not be able to log in on an iOS device, and vice versa. This problem was solved by storing the encrypted password encoded as a Base64 string, and using this format when comparing, since it is independent of the platform. This problem was discovered during early development and solved before the structured test (Section 4.1.1).

The second problem was the interaction flow associated with user login and sign-up. Figure 3.3 shows the two screens dedicated to authentication: the login screen and the sign-up screen. In the login screen, there was a button to create a new account, leading to the sign-up screen. When the users entered the sign-up screen, an alert appeared, informing that only mentors should create an account through that screen. The mentor could then fill in their personal information and create an account. After the mentor logged in, they could add mentees by IST ID. After adding mentees, the mentees should create their accounts using the login screen (not the sign-up screen). In the login screen, there was an input field for users to type the IST ID. After doing so, the application would verify if the IST ID matched with that of an already created account – if so, the user could then type their password and log in. If it did not match, the application would verify if it matched an IST ID that was added by a mentor (i.e. it was a mentee’s account not yet created), and the user would be prompted to fill in their personal information to create an account. This flow caused a lot of confusion during the structured test (Section 4.1.1), because mentees often dismissed the alert without reading it and ended up creating accounts as if they were mentors.

Additionally, all the authentication was done locally on the mobile device, because in the old architecture (Figure 2.1) the server only hosted the CouchDB database, there were no auxiliary services. The user passwords were generated (on sign-up) and verified (on login) on the mobile devices. This meant that the encryption algorithm ran on the mobile devices, and that the encrypted passwords were stored in the user document, which the local database replica would then sync to the remote server database. There were two problems with this approach. Firstly, for the encrypted key to be secure, PBKDF2 has to run many iterations (it ran five thousand) and this caused a noticeable delay when signing up or logging in. Secondly, the encrypted password (and the cryptographic salt that was used to generate it) should
be stored securely on the server, instead of "travelling back and forth" between the server and mobile devices – this will happen countless times due to database replication, and it is even more problematic because each user document is shared between mentor and mentee(s).

Redesigning the interaction flow (V2)

Given the problems of V1, it was necessary to redesign the interaction flow of the authentication process before the start of the unstructured test (Section 4.1.2). The aspect that caused the most confusion in students was that mentees were not supposed to use the sign-up screen to create an account. To simplify the process, a more familiar pattern was chosen (Figure 3.4, left) with an entry screen that has two buttons: one that leads to the login screen and another that leads to the sign-up screen. Along with the redesign of the interaction flow, V2 also introduces some new application logic to the authentication process. An aspect that was not handled by V1 was authorisation – previously, any new user could sign up, and an existing user could log in at any time. A new requirement was to be able to control who can use the application, so that only users who are part of the MP-TP are allowed access to the mobile application. Fulfilling this requirement was made easier by the addition of the roles field to the user document. When users type their credentials in the login screen (Figure 3.4, right), the application will not only verify that they match with the credentials of an existing account, but also that the account has a role associated with the current academic year. This makes it possible to ensure that students who are no longer part of the MP-TP cannot access other users’ information, such as that of previous mentor or mentees.

This authorisation aspect was also applied to user sign-up. When users press the sign-up button on the entry screen, they are shown a screen where they can type their IST ID. The application will then verify if the IST ID is valid – that is, if it is the IST ID of a new mentor in the current academic year, which would have been previously added by MP-TP’s coordinator using the Web application; or the IST ID of a new mentee, which would have been added by the student’s mentor using the mobile application. If the IST ID is deemed invalid, the user will be denied access and shown an error message (Figure 3.5, left). Otherwise, the user will be shown a screen where they can fill in their personal information and create
an account (Figure 3.5, right). The application will also recognise if the IST ID corresponds to an existing account, and inform the user that the account already exists.

Finally, it should be noted that in V2, all the authentication and authorisation logic is still done locally by the mobile application. Since it had already been decided that NAPP should ultimately use FenixEdu’s Central Authentication Service\(^3\), V2 was meant to be an intermediate solution that only intended to solve the previous interaction flow issues and to test the new authorisation logic in the unstructured test.

\(^3\)FenixEdu is an open-source academic information platform created by and used at Técnico Lisboa. \url{http://fenixedu.org}
FenixEdu Authentication (V3)

The sole focus of V3 was to migrate NAPP's authentication logic to use the Central Authentication Service provided by FenixEdu. Not only is this more convenient for students, who will not have to memorise a new set of credentials to use NAPP, but it will also allow to integrate NAPP with the FenixEdu API in the future.

In V3 (Figure 3.6), users log in with their Técnico Lisboa credentials. The mobile application will open a Web page (hosted by NAPP) that redirects them to FenixEdu's authentication page. Firstly, users will be asked to authorise FenixEdu to provide their information to NAPP. If users grant permission, they can log in with their credentials. After the login, the Web page makes a request NAPP's REST API, which contacts FenixEdu to get the user's access token. When token is received by the REST API, it is sent back to the Web page and the token is saved in the mobile application. This token can be used to make requests to the FenixEdu API. At this point, the user is authenticated, and the authorisation phase begins. Using the access token obtained during authentication, the mobile application will make a request to the FenixEdu API to get the user's personal information. From this information, the mobile application takes the user's IST ID, and makes a request to NAPP's API to verify that the user is authorised to use NAPP. The authorisation criteria of V3 are very similar to those of V2, but now the authorisation code runs on the server. With the user's IST ID, the server code will access the Users database and grant permission based on the following criteria:

- If a user account exists and the user has a role in the current academic year, the user is authorised. The user can then proceed to use the mobile application.
- If a user account exists but the user does not have a role in the current academic year, the user is denied access. The mobile application will show an error message.
- If a user account does not exist yet, but the IST ID provided matches with that of an enrolled mentor or mentee, the user is authorised. The user will fill in their personal information in the mobile application and can then proceed to use it.
- If a user account does not exist and the user is not enrolled, the user is denied access. The mobile application will show an error message.
Reporting of grades

As mentioned before, one of the biggest problems with the NAPP framework was that the reporting of grades was not integrated with the evaluation methods of the courses. With the improvements made to the data model, the mobile application could then be modified so that the grades reported by mentees are associated with evaluations. Figure 3.7 shows part of the interaction that will take place when reporting a grade. The user must choose a course and an evaluation in order to report a grade. The courses presented to the user are retrieved from the Courses database, based on the current academic year and semester, and on the user’s degree. When the user types the grade, the application will also verify that the grade is within the limits ($minGrade$ and $maxGrade$) for the chosen evaluation, and show an error message in case it is not (Figure 3.7, right).

Other improvements

Besides the improvements explained in the previous sections, other smaller improvements were made to the mobile application.

The logging of user actions was implemented, and a document is added to the Log database after a relevant action is completed.

The Pomodoro timer was enhanced to work when the application is in the background – when the user locks the mobile device or navigates away from the NAPP application, the timer will keep running and a notification will appear on the notifications panel informing that the timer is still active.

Input validation was added to every input field where the user input must follow certain rules, and feedback is given when there are errors. This included not only validating that the grade is within limits when reporting, but also that the user information on sign-up is correct, and that when mentors add new mentees the IST ID follows the expected pattern. Input validation ensures the data added by the users are correct, and giving feedback when there are errors contributes to a better user experience.

Further enhancements concerned database replication. CouchDB and PouchDB allow to choose
the replication direction of each database when initiating the sync: from the local replica to the remote server database, from the remote server database to the local replica, or bidirectional replication. For simplicity, when talking about replication in the previous sections, it was implied that all the databases used bidirectional replication. However, these replication directions can be leveraged to improve the performance of the mobile application, and avoid wasting resources and storage, because not all the databases need to sync bidirectionally. From the perspective of the mobile application, the database replication directions chosen were the following:

- The Users database syncs bidirectionally, because a given user’s document is accessible to more than one client application (e.g. if the mentee modifies their user document, the mentor must see this change reflected, and vice versa),
- The Tasks database syncs bidirectionally, because tasks assigned by a mentor must appear in the mentee’s task list (hence, from the local mentor replica to the remote server database, and from the remote server database to the local mentee replica)
- The Feedback database syncs only from the local replica to the remote server database, because the history of feedback submitted is not shown in the mobile application,
- The Courses database syncs only from the remote server database to the mobile application, because course information is not modified in the mobile application,
- The Log database syncs only from the local replica to the remote server database, because the history of logs is not shown in the mobile application.

Besides replication directions, CouchDB and PouchDB allow further configurations, such as filtered replication. By default, database replication syncs all the documents in the database, but it is possible to filter which documents to replicate. It should be noted that filtered replication is not the appropriate mechanism to use in order to implement access control and restrict access to documents, and that it is used in the mobile application to reduce the amount of data stored in the mobile device. In the mobile application, database sync begins only after authentication. On a successful login, the mobile application obtains the _id of the user document. The _id is then used to filter the Users database so that only the relevant user documents are synced to the local replica. After syncing the user’s document, the roles field is used to determine what other documents should be synced – if the user is a mentee in the current academic year, their mentor’s document should also be synced; if the user is a mentor in the current academic year, their mentees’ documents should also be synced. Likewise, the Tasks database is also filtered so that only relevant tasks are synced. The author and target fields of the task document are used for this purpose. The Tasks database will only sync the documents where the value of the author or target fields is equal to the _id of the user – this way, mentees will have access to their tasks (the mentee is the author) and to the tasks assigned by their mentors (the mentee is the target), and mentors will have access to the tasks which they assigned to mentees (the mentor is the author).
3.3 The New Web Application

3.3.1 The Courses feature

The Courses feature gives the MP-TP’s coordinator the ability to manage courses’ information, and to visualise and analyse the results of each course’s evaluation. The Courses feature is comprised of three pages: the list of courses, the course page, and the page to add a new course.

The list of courses

This page displays a list of cards, each representing a course (Figure 3.8). Each card has a hyperlink to the course’s page, and also shows basic information about the course, such as the evaluation names and their dates. Additionally, each card has two buttons. The first leads to a page that allows to edit the course’s information. The editing page is in every way identical to the page for adding a new course (explained below). The changes made to the course are reflected in the course document in the Courses database. The second button allows to remove the course – a dialogue is shown first, and after user confirmation, the course document is deleted from the Courses database. Above the list of courses, there are controls which allow to filter the list by degree, academic year and semester. The courses are displayed in alphabetical order.

![Figure 3.8: Web application – the list of courses](image)

The course page

The course page is the most important page in the Courses feature. The main goal of this page is to facilitate the analysis of students’ results for each evaluation of the course.

As mentioned in the Introduction, without NAPP’s Web application, the MP-TP’s coordinator makes use of a very basic spreadsheet to analyse each student’s academic results. The coordinator reported that the spreadsheet is not configured to generate evaluation statistics, because of a lack of advanced spreadsheet manipulation skills from the coordinator’s part. However, the coordinator expressed the wish to have some statistical analysis be done automatically by the Web application. As such, the requirements for the course page were the following:
The page must provide a way to quickly identify the students who had negative grades in an evaluation.

The page must automatically calculate basic statistics of an evaluation (such as average grade, number and percentage of negatives and positives).

Near the top of the course page (Figure 3.9), there are two filter controls which determine the remaining content of the page. One of the controls selects the evaluation for which to present the results (the evaluation is selected from the evaluationMethod of the course). The second control filters the results by academic degree. The evaluation results are divided into two main sections: the distribution of grades and the statistics.

The distribution of grades is comprised of a bar chart and a table. On the chart, the $x$ axis shows all possible integer values from minGrade to maxGrade inclusive. On the $y$ axis, each value represents the number of students who obtained the grade at $x$ (the students’ grades are rounded to the nearest integer). The table below the chart shows the list of students and their grades. The table can be sorted by any of the columns, either in ascending or descending order. The default sorting order is by grade, ascending – so that students with lower grades appear at the top. The chart is interactive, and the contents of the table depend on the interactions with the chart. At first, the table shows all the students who reported their grades. When a bar is clicked, the table will be filtered to only show the students who obtained the grade relating to the selected bar. Moreover, when hovering over a bar of the chart, a small panel appears, showing the label for that bar.

The statistics section has a circular chart that shows how the students are partitioned in terms of number of positive, negative, and not reported grades. The chart also shows labels when each section is hovered over. Additionally, the same statistics are presented in text format below the chart, along with the average grade for the evaluation.

Adding a new course

This page is used to create a new course, by filling in the information necessary, as specified by the data model. The page contains a form (Figure 3.10) that is divided into two parts: first the user inserts the basic information about the course, then the user adds the evaluation method. There are field validation rules in place, and error messages are shown to the user if any field does not follow the rules. This validation includes: checking that all required fields are filled, and that the information inserted is valid (e.g. the academic year is valid, the evaluations’ grade limits and weight are valid).

At this point, without integration with the FenixEdu API, all the information regarding each course must be inserted manually by the MP-TP’s coordinator.
Figure 3.9: Web application – the page of the “Álgebra Linear” course, showing information about one of the course’s evaluations
Figure 3.10: Web application – adding a course requires filling in basic information (top) and then the evaluation method (bottom)
3.3.2 The Students feature

The Students feature allows the MP-TP coordinator to view and manage information related to the students who take part in the mentoring programme. Most importantly, the coordinator is able to follow each student's academic progress. Throughout the following sections, three terms are used to describe the status of students within the mentoring programme: enrolled, registered and unregistered.

- **Enrolled** students are those who have been admitted to the programme (mentors are enrolled by the coordinator through the Web application, mentees are enrolled by their mentors through the mobile application).

- **Registered** students are those who are enrolled and have completed sign-up using the mobile application.

- **Unregistered** students are those who are enrolled but have not signed up using the mobile application.

The Students feature is comprised of four pages: the students list, the student page, the page to enrol mentors, and the page to view unregistered students.

The students list

The students list (Figure 3.11) displays the registered students who take part in the MP-TP, and it also provides ways to directly access students’ pages. At the top of the page there is a search bar, where the user can lookup a student by name (firstName or lastName) or by student number (studentNumber), and access the student's page. The page also displays the list of all registered students, with mentees on the left side and mentors on the right side. Each entry in the list features a hyperlink to the student's page. Two controls can be used to filter the list by academic year and degree. The list can be ordered by student name or number. Below the controls, there is a hyperlink that leads to the page for viewing the unregistered students in the selected academic year.

![Figure 3.11: Web application – the students list](image)
The student page

The goal of the student page is to allow the MP-TP’s coordinator to follow a specific student’s progress. In the case of mentees, this refers to the academic progress; in the case of mentors, it concerns the progress within the mentoring programme. The student page has two possible views, depending on the role of the student: the mentee view and the mentor view.

During development, the coordinator explained that, to follow the progress of a mentee, they would use a colour code and apply it on the spreadsheet that keeps the reported grades. Each line on the spreadsheet refers to a mentee: if the student had reported one or two negative grades, the line would be coloured yellow; if the student had reported more than two, the line would be coloured red. This was a manual, time-consuming process. As such, the requirements for the mentee view were the following: it must facilitate the identification of negative grades; it must also provide a way to visualise the student’s progress over time. As for the mentor view, there were no initial requirements. During the development stage, the coordinator expressed an idea to create a points system to evaluate mentors. This idea came up very late in the development stage, and thus there was no time to formally define it. For this reason, the mentor view is lacking in functionality. As a general requirement of the student page, not initially tied to any of the two views in particular, the student page should allow to switch the mentor of an existing mentee in the current academic year.

The student page shows detailed information about the student. Below the header (which shows the student name and number, and picture place-holder), there is a control where the user can select the user role (linked to an academic year) for which to view details. Upon selection, the page will switch between views.

The mentor view (Figure 3.12) has very little functionality at the moment. There are only two tabs. The Personal data tab shows the student’s personal information. The Mentees tab lists the mentors’s mentees (both registered and unregistered). For the registered mentees, the list has a hyperlink to each mentee’s page, next to a summary view of that mentee’s academic performance.

The mentee view has four tabs. The Personal Data tab shows the student’s personal information. The Grades tab (Figure 3.13) can be used to analyse the mentee’s academic performance. The Grades tab shows, for each semester, a line chart that plots the grades reported by the student. The chart allows for easier visualisation of the evolution of the student’s grades over time. The x axis displays dates
The y axis ranges between 0% and 100%. Since the grade intervals (minGrade and maxGrade) of each evaluation are variable, it was necessary normalise the grades to the same range, and the 0% to 100% range was chosen. To plot a grade on the chart, the x coordinate will be the date of the evaluation. If the evaluation does not have a date defined, the x coordinate will be the date in which the grade was reported. The y coordinate will be the value of the grade normalised to the 0% to 100% range. The chart's points and connecting line are coloured as a gradient that spans red (0%) to green (100%). The chart is interactive: when hovering over a point, a small panel will show details for the pointed grade, including the evaluation name and course, the student's grade, and the grade interval. Below the chart, the page also displays tables, where it is possible to view the student's grades in every evaluation of each course. The Study Tools tab (Figure 3.14) presents an overview of the use of the tasks list feature in the mobile application. It shows a stacked bar chart with two categories: tasks assigned by the mentor, and tasks created by the mentee. Each category is then divided into two groups: pending tasks, and completed tasks. Lastly, the Mentor tab shows a hyperlink that leads to the student page of the mentee's mentor. If the mentor-mentee pairing refers to the current academic year, there is also a button on the right-hand side that allows to switch the mentee's mentor (Figure 3.15).

---

4For better visualisation, the time scale on the x axis adjusts to the grades reported at the time of viewing (as can be seen on Figure 3.13, which displays only the month of September). The x axis time scale is configured to always fall between the approximate start and end dates of the semester.
Enrolling new mentors

Another page in the Students feature allows the coordinator to enrol new mentors (Figure 3.16). The enrolment of new mentors relates to the new authorisation mechanisms explained in Section 3.2.2: mentors must be enrolled beforehand, in order to be able to sign-up using the mobile application.

To enrol new mentors, the coordinator must insert an academic year and a list of mentors. Each entry in the list of mentors corresponds to the IST ID of the student. For convenience, instead of having to type out the IST IDs of the students, the coordinator should submit an Excel file. There are instructions on screen on how to correctly create the Excel file. The Web application then reads the Excel file and makes a request to the REST API, to enrol the mentors whose IST IDs were present in the file.

Viewing unregistered students

The last page of the Students feature allows to view the unregistered students in a given academic year (Figure 3.17). The left column shows the IST IDs of mentors who have been enrolled by the MP-TP’s coordinator but are not yet registered. The right column shows the IST IDs of the unregistered mentees grouped by mentor, so that the coordinator can easily reach out to the mentors who still have unregistered mentees.
Figure 3.15: Web application – the Mentor tab of the mentee’s view (top) which allows to switch the student's mentor (bottom)
Figure 3.16: Web application – the page to enrol mentors in the MP-TP

Figure 3.17: Web application – the page to view unregistered mentors and mentees
3.3.3 The Alerts feature

The objective of the Alerts feature is to deliver alerts about relevant events within NAPP to the Web application. At this point, the only event that was identified as being valuable is when a student reports a negative grade. Thus, the Alerts feature serves the purpose of helping the MP-TP’s coordinator to identify at-risk students.

The Alerts feature was designed with extensibility in mind, so that, in the future, it can support different types of alerts, and not only alerts relating to the report of negative grades. The goal is to give the coordinator the ability to customise, within a set of predefined parameters, what types of alerts they would like to receive. For this purpose, it was necessary to create and model an entity that can describe the preferences of the coordinator – this entity was named AlertsPreference. Listing 3.6 defines the structure of the AlertsPreference entity. The *name* field is the “user-friendly” name for the preference. The *students* field defines which students the preference applies to – the possible values are "all" (which means it should apply to all students) or an array with the _ids of the students. The *event* field must correspond to one of the possible *actionIDs* of a log document (see Table 3.8). The *condition* field refers to a condition that the log entry must meet. This field is an object that contains a field named *type*, which is simply a code to describe the condition.

```typescript
interface AlertsPreference {
  name: string;
  students: "all" | string[];
  event: string; // must match actionID
  condition: {
    type: string;
  }
}
```

Listing 3.6: The structure of the AlertsPreference entity, in Typescript notation for declaring object interfaces

As an example, Listing 3.7 shows the only preference that is defined for the coordinator at the moment, referring to the report of a negative grade. This preference concerns all students, and refers to the "reportGrade" event. The condition is identified by the code "is-negative", which is a predefined type recognised by the Alerts feature.

```json
{
  "name": "Aluno reporta nota negativa",
  "student": "all",
  "event": "reportGrade",
  "condition": {
    "type": "is-negative",
  }
}
```

Listing 3.7: The AlertsPreference that will trigger an alert when a student reports an negative grade, in JSON notation

The coordinator’s preferences are kept in a document in the Users database – the coordinator’s document. Although it is saved in the same database as the students’ documents, the coordinator’s document does not follow the same structure. Table 3.10 shows the structure of the coordinator’s document. The value of the _id field is currently "coordinator". In the future, when FenixEdu authentication is implemented in the Web application, the _id will be the IST ID of the coordinator. Besides the preferences,
two other fields are saved in the document: the alerts field and the token field.

The alerts field stores the alerts that are generated. Each alert has the structure defined in Listing 3.8. An alert has a title and optionally a subtitle, it has a description and the timestamp of the moment when it was generated. The value of the actionUrl field is a hyperlink to the Web page that should be opened when the user clicks the alert.

```
interface Alert {
    timestamp: string;
    title: string;
    subtitle?: string; // optional
    description: string;
    actionUrl: string;
}
```

Listing 3.8: The structure of the Alert entity, in Typescript notation for declaring object interfaces

The token field relates to Firebase Cloud Messaging (FCM). The Alerts feature uses FCM to deliver the alert as a push notification, which appears in the coordinator’s device even if they are not using the Web application at that time. The coordinator has to allow the Web application to show push notifications to activate this functionality, and they can opt out at any time – the alerts will still be delivered inside the Web application even if the coordinator has opted out of push notifications. The token field is sent to FCM along with the alert, and it is used by FCM to identify the device to which the push notification should be sent.

As illustrated in the system architecture (Figure 3.1), the Alerts feature runs server-side. It is a programme that runs continually on the server. The program syncs with the Log database and listens to changes, so that every time a new document is written to the database, it will trigger the execution of some procedure associated with the Alerts feature. This procedure determines whether the event associated with the new log document should cause an alert to be delivered to the MP-TP’s coordinator, and it also handles the delivery of the alert.

Algorithm 3.1 describes this procedure in high level. The the new log document is received as input, and the first thing to do is to read the coordinator’s document from the database (line 1). Then, the log will be processed according to each of the coordinator’s preferences, to determine which ones should trigger alerts. For this purpose, a handler is created for the preference currently being analysed (line 3). The handler specialises in a type of preference, and can process a given log document according to that preference. The creation of the handler is done by analysing the preference in a tree-like manner: first, the root node examines the event field, and dispatches the creation to the appropriate node in the second level (the event-specific level); then, that second-level node examines the condition field and creates the handler. As an example, let us consider the creation of the handler for the preference shown in Listing 3.7. The root node examines the event field, whose value is "reportGrade",
Input: log

1. coordinator ← read coordinator document from the database
2. foreach preference of coordinator do
   3. handler ← create handler for the preference
   4. result ← handler processes the log
   5. if result ≠ null then  // the log matches the preference
       6. alert ← handler generates alert from the result
       7. write alert in coordinator document in the database
       8. if coordinator allowed notifications then
           9. send alert as FCM notification
       end
   end
end

Algorithm 3.1: The processing of a log entry to deliver alerts

dispatches the handler creation to the "reportGrade" node in second level, a node that specialises in creating handlers for "reportGrade"-related preferences. Then, the "reportGrade" node examines the condition field, which has the type "is-negative", and creates the handler that specialises in the preference.

After the handler is created, it is time to make the handler process the log (line 4). At its core, this processing compares the log with what the handler expects. Back to the example of the preference shown in Listing 3.7, the handler for this preference verifies that the istID of the log fits the students value of the preference (in this case, it always does because the value of students is "all", so any istID should fit); then it verifies that the actionID of the log is equal to the value of the event field of the preference (which is "reportGrade"); and lastly, since it has confirmed that the log refers to a grade report, the handler will check that the grade associated with the log (in the log's detail field, see Table 3.9) is actually a negative grade – this will require accessing the Courses database and comparing the student's grade with the evaluation's minGrade and maxGrade.

If any of these verifications fails during processing, the handler returns a "null" value, and an alert is not generated. If, however, the result of the processing is not "null" (line 5), it will be a value that the handler can use to generate an alert (line 6). The generated alert, is then saved to the coordinator's document in the database (line 7). If the coordinator has allowed notifications to be delivered (line 8), the alert will be sent to FCM (line 9), which will take care of delivering a push notification to the device of the MP-TP's coordinator.

Whether an alert for the current preference was generated or not, the procedure will move on to processing the log according to the next preference (goes back to line 3), until all of the coordinator's preferences have been analysed. This makes it possible for a single log document to generate many alerts, if it matches with more than one preference.

This layered architecture of creating preference handlers that specialise in processing a log according to a particular preference is what makes it easier to extend the alerts feature to other types of preferences. This architecture makes the source code modular, with each type of handler being a plug-in component that has the single responsibility of handling a specific preference.

Finally, in the Web application, the list of alerts is shown in a panel that overlays the current page (shown in Figure 3.18 covering part of the page to enrol mentors). The alerts panel can be opened by clicking the "bell" button on the top-right corner of the screen. The alerts in the list will lead to a page of the Web application when clicked – in the case of the negative grades alerts, the student's page.
3.3.4 The Dashboard feature

The goal of the Dashboard is to allow the coordinator to view relevant, up-to-date information about the MP-TP without the need to browse through the other pages of the Web application. This information is conveniently aggregated and presented in the form of “cards” in the Dashboard page (Figure 3.19). The information presented in the Dashboard page includes:

- A table with statistics about evaluations that took place within the last month. For every recent evaluation, the card shows how many grades have been reported, how many of those are negative grades, and the average grade. Clicking an evaluation in the table will lead to the course’s page, where more detailed information can be explored,

- A table with the dates of evaluations that will happen within the next two weeks,

- A list of at-risk students, featuring students who have reported at least one negative grade. The items on this list can be expanded to show the list of negative grades of the student and a hyperlink to the student’s page.

Other important information that is presented in the Dashboard is related to the use of NAPP’s mobile application. One card shows the number of enrolled versus unregistered mentors and mentees. This is particularly relevant at the start of the academic year, so that the coordinator can keep track of which students have not yet created an account in the mobile application. Another card provides a visualisation of the activity logs generated by the students using the mobile application. The card features two charts, which display the usage over the past week and month. Below the charts there are buttons that can be used to filter the data in the charts by type of log: the categories are total, tasks, Pomodoros, grade reports, and suggestions.

As this is a Dashboard, it is important that the information is up-to-date, so the page is configured to refresh every five minutes. At the top of the page there is a section that shows how long ago the page was updated and it also has a button that will force the update. Additionally, all the information that is presented in the Dashboard is requested to the REST API, because the processing of the “raw” data, stored in the databases, into useful information is generally a CPU-intensive task. This makes the processing unsuitable to run on client machines, not only because it can slow down the client machine,
but also because these kinds of machines often do not have the computational resources needed to execute this processing in due time.

### 3.3.5 The Feedback feature

The Feedback feature is made up of a single page where the MP-TP’s coordinator can view the suggestions submitted by mentees. The suggestions are sorted by date, showing newest suggestions first. Above the list there is a search box that can be used to search for words within the text of the suggestions – as pictured in Figure 3.20, where only suggestions containing the word “actividades” are shown.
The REST API currently serves three purposes within NAPP. Firstly, it is used for the authorisation phase of Authentication V3 (see Section 3.2.2). Secondly, it is used by the Web application to request information that is not fit to be calculated in the Web application, because it results of CPU-intensive operations. Lastly, it carries out write operations to the Users database on behalf of the Web application, such as enrolling new mentors or switching the mentor of a mentee.

Table 3.11 describes the API endpoints and where they are used within NAPP. The endpoints provided by the REST API can be divided into four categories:

- **Authentication**: endpoints relating to the authentication process
- **Academic**: endpoints to retrieve information related to school, such as evaluations or student grades
- **Analytics**: endpoints to retrieve statistical information about the usage of NAPP
- **Mentoring**: endpoints to retrieve or update information about the MP-TP
<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint details</th>
<th>Used in</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Path: /academic/students/grades</td>
<td>Student page (Grades tab of mentee’s view)</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>userId (the _id of the student)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>academicYear</td>
<td></td>
</tr>
<tr>
<td></td>
<td>semester</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the student's reported and not reported grades for the given academic year and semester, grouped by course; as well as the evolution of the student's academic performance (reported grades sorted by time)</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /academic/students/at-risk</td>
<td>Dashboard</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>academicYear</td>
<td></td>
</tr>
<tr>
<td></td>
<td>semester</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the students who have reported at least one negative grade in the given academic year and semester, along with their negative grades</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /academic/evaluations/upcoming</td>
<td>Dashboard</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the upcoming evaluations happening within the next two weeks (according to server time)</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /academic/evaluations/stats</td>
<td>Dashboard</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the statistics of evaluations that happened within the past month (according to server time)</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /analytics/mentee/tasks</td>
<td>Student page (Study Tools tab of mentee’s view)</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>userId (the _id of the student)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the mentee’s usage statistics for the tasks list feature of the mobile application (i.e. how many tasks were created by the mentee and assigned by the mentor; how many of these are pending and completed)</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /analytics/general/usage</td>
<td>Dashboard</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the students’ usage statistics for the mobile application – total and by log actionID – for the past month (according to server time)</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /analytics/general/registrations</td>
<td>Dashboard</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>academicYear</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>get the statistics of enrolments and registrations of mentors and mentees, for the given academic year</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Endpoint details</td>
<td>Used in</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>POST</td>
<td>Path: /mentoring/pairings/switch-mentor</td>
<td>Student page (Mentor tab of mentee’s view)</td>
</tr>
<tr>
<td>Body (JSON):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mentee: &lt;MENTEE_ID&gt;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mentor: &lt;MENTOR_ID&gt;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>academicYear: &lt;ACADEMIC_YEAR&gt;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>switches the mentee’s current mentor for the provided mentor, in the given academic year</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>Path: /mentoring/pairings/enrol-mentors</td>
<td>Page to enrol new mentors</td>
</tr>
<tr>
<td>Body (JSON):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>academicYear: &lt;ACADEMIC_YEAR&gt;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mentors: [&lt;MENTOR_ID_1&gt;, &lt;MENTOR_ID_2&gt;, ...]</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>enrolls the mentors in the given academic year; returns the result of the operation: _ids of mentors enrolled, _ids of mentors previously enrolled (conflicts), duplicate and invalid _ids given</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /mentoring/unregistered-students</td>
<td>Unregistered students page</td>
</tr>
<tr>
<td>Parameters:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>academicYear</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>get the list of unregistered mentors and mentees (mentees are grouped by mentor) in the given academic year</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Path: /auth/get-fenix-token</td>
<td>Mobile application</td>
</tr>
<tr>
<td>Parameters:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>code (FenixEdu-related)</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>get the access token associated with the given user code from the FenixEdu API. This is part of the Authentication phase of Authentication V3 (Section 3.2.2)</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>Path: /auth/authorise-student</td>
<td>Mobile application</td>
</tr>
<tr>
<td>Body (JSON):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>id: &lt;IST_ID&gt;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>This is part of the Authorisation phase of Authentication V3 (Section 3.2.2)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.11: The REST API endpoints
3.4 Deployment

As of October 2018, the licences to publish NAPP’s mobile application on “app stores” – Google Play (Android) and iOS App Store (iOS) – have been acquired, and the mobile application will be published before the start of the tests with experimental and control groups.

The Server pictured in the system architecture (Section 3.1.1) will actually be deployed as two separate servers. These servers are hosted by Técnico Lisboa’s virtualisation platform, powered by OpenStack software. Given the resources provided (16GB of RAM, 8 Virtual CPU, and 100GB of storage) and the system configurations available in the platform, it was necessary to divide the server into two virtual machines, each with half the resource quotas. The first virtual server runs CouchDB exclusively, thus becoming the dedicated database server. The second virtual server runs the REST API, the Alerts worker, and also Apache Web Server software which serves the Web application and the Web page used in Authentication V3. Each virtual server has a public IP address, making it accessible from outside of Técnico Lisboa’s campi. Both servers have been configured with HTTPS for secure communications, with SSL certificates provided by Técnico Lisboa.

Finally, a Firebase developer account was created on behalf of NAPP with the free plan, which includes unlimited access to Firebase Cloud Messaging.
Chapter 4

Evaluation

4.1 Beta testing the mobile application

4.1.1 Structured test

Since NAPP’s mobile application was initially developed as a prototype, it had never been used in any situation other than the very controlled environment of usability testing. It had never run on a real smartphone, nor connected to a database that was not hosted on a personal computer.

The first objective was to test the mobile application on real mobile devices to see how it would function when it was being used simultaneously by multiple users. In order to test the application on real mobile devices, the ideal course of action was to test it on both Android and iOS devices. At that point, the acquisition of the licences for publishing the application on “app stores” had not yet been authorised by Técnico Lisboa’s administrative services, so it was necessary to turn to an alternative course of action: distribute the application, to the users who would participate in the test, outside of official channels.

Regarding iOS devices, the only way to install an unpublished application to the devices is to start an official beta test, which relies on software named TestFlight\(^1\). However, the steps to follow in order to start a beta test with TestFlight are essentially common to those necessary for publishing the application to the iOS App Store\([18]\), and as such, a licence is still required. On the other hand, Android makes the installation of unpublished applications much easier: one only needs to build the mobile application into an Android package (.apk extension), which can be installed on any Android device, without the need for a license for distribution through Google Play. Therefore, while the licences were not acquired, it was only possible to test NAPP’s mobile application on Android devices.

At the time of this test, the access to Técnico Lisboa’s virtualisation platform had also not been provided. For this reason, CouchDB was installed on a desktop computer inside Técnico Lisboa’s Taguspark premises, and connected to the local network – this computer became the Server during the test. The idea for this test was to have the users follow a guide with specific chores planned in advance. Because of the limitations of the Server, users were instructed to complete the chores while they were on campus.

The chores on the guide were intended to thoroughly test the possible scenarios related to the features of the mobile application, to verify:

- How the tasks list behaved with concurrent online and offline writes,

- If the Pomodoro timer worked as expected, without problems,

- If grades reported by a mentee while offline could be viewed by the mentor when the mentee’s device was online again,
- If mentees were able to submit feedback.

Given these objectives, a test guide was devised (Appendix C). The guide consisted of chores to be executed by the users through the course of three days – the test took place from the 1st to the 3rd of March, 2018. The guide emulates scenarios of interaction between two mentors and their mentees. The first mentor had four mentees, and the second mentor had two mentees – in total, eight users participated in the tests. All users were given a copy of the guide appropriate for their role in the test. Each day had a chore to be executed in the morning and another to be executed in the afternoon. Along with the description of each chore, the expected result of the chore was explained, and the users were asked to write down if the result they observed matched the expected result. Additionally, to properly test the offline behaviour of NAPP, the second day featured chores that were to be executed offline, while the chores of the first and third days were to be executed online.

This test allowed to identify the following problems with NAPP:

- The replication conflicts occurring from the centralisation of all the user-related data in a single document – which lead to the creation of the Tasks database,
- The Pomodoro timer would stop working as soon as the user left that screen or locked the device – this was later fixed, and the timer now works throughout the whole application, as well as in the background when the user leaves the application or locks the mobile device,
- Users had trouble understanding the interaction flow of the user login and sign-up, which resulted in some mentees accidentally creating mentor accounts – this prompted the redesign of the authentication’s interaction flow (V2).

Overall, this was a very important test because it allowed to diagnose two critical problems in NAPP that could not have been discovered with the application running on a local environment, and a problem with the mobile application’s User Interface.

### 4.1.2 Unstructured test

After correcting the problems that were found during the structured test, the next objective was to test the mobile application in a real environment over the course of nearly the entire semester, to be able to collect the data needed to evaluate the value of NAPP to the MP-TP. However, at that time, the acquisition of the licences for publishing the mobile application had still not been approved, so there was still the constraint that the mobile application could only be installed on Android devices.

An online form was created for students to sign up for the test, and a short presentation was given at the end of two classes, to try to gather mentees to participate in the test. During these presentations, most of the mentees who showed interest happened to have iOS devices. Mentors were also informed of the test, and were encouraged to participate with their mentees. Approximately 30 mentors and mentees signed up for the test – yet, some mentees signed up whose mentors had iOS devices, and vice versa. Since the goal was to run the test in a real environment, with mentees paired up with their actual mentors, it was not suitable to pair up these remaining students. In the end, 5 mentors and 11 mentees participated in the test.

Because this was a very small number of students, it was not possible to do any meaningful statistical analysis about the use of NAPP’s mobile application. In the mean time, since the end of the structured test, NAPP had been granted access to Técnico Lisboa’s virtualisation platform. This meant CouchDB
Given these circumstances, it was decided that the test would still be executed but the objectives had to be redefined. The objective of this test was no longer to collect data to evaluate the value of NAPP to the MP-TP, but rather to get a first impression of how much the students would use the mobile application over an extended period of time.

The test took place between April 10th and June 26th, 2018. Over the course of nearly three months, the students generated 132 activity logs. Figure 4.1 shows the distribution of activity logs over time. It is possible to see that there was higher activity during the first week and that later it slowed down. During the remaining time, students mostly reported grades and used the tasks list tool. As can be seen in the distribution, although there were peaks in activity, students used the mobile application at least once a week (except for the week between May 29th and June 5th). In a feedback session after the test ended, students reported that what they struggled the most with was knowing when the mentor interacted with them, or when the mentee reported a grade or completed a task – this difficulty to remain engaged with the mobile application explains the peaks in activity seen in the distribution. The students called for a notifications system that would let them know of these relevant actions. Other prevalent suggestions amongst the students were to add functionality to the mentor’s view in the mobile application and to improve the visual appearance of the mobile application, which they considered too minimalistic.

### 4.2 Web application usability test

In order to be able to detect flaws in its design, the Web application needed to undergo usability testing. In a usability test, the users are asked to perform a set of tasks using the system. These tasks should represent a realistic scenario of usage. In this test, the Courses and Students features were tested individually, but the Dashboard and Feedback features were tested as a group. The decision of grouping these two features was based solely on the small number of tasks for them. The Alerts feature was not tested because of the lack of meaningful tasks concerning this feature.

It is important to note that the purpose of these tests was to evaluate the Web application’s features
from a qualitative viewpoint: the goal was to gain insight into common errors, to be able to correct them. The purpose of this test was not to do a quantitative usability study by analysing metrics such as user efficiency, number of user errors or subjective satisfaction. As advocated by Nielsen, when running a usability test that is not meant for a quantitative usability study, testing five users almost always reaches the maximum benefit-cost ratio of user testing.[19, 20] For these reasons, the Courses feature was tested with seven users and the remaining features were tested with five users.

The tasks created for the purpose of testing the Courses feature were the following:

1. **Add a new course** – given the course information in natural language,
2. **Edit a specific course** – given the information to edit in natural language,
3. **Remove a specific course**,
4. **Filter the list of courses to only see courses taught to the “LEIC” degree in the first semester of the academic year 2017/2018** – users had to write down how many courses the list had after filtering,
5. **Access the course page for “Linear Algebra”**,  
   5.1. **What percentage of students did not report grades in the first test?** – the goal was to check if users could interpret the evaluation statistics,
   5.2. **How many students obtained a grade of approximately 14 in the second test?** – the goal was to check if users could read the bar chart that shows the distribution of grades,
   5.3. **How many students from the “LEIC” degree obtained a positive grade in the first test?** – the goal was to check if users could filter the information by academic degree,
   5.4. **What is the IST ID of the student who obtained the lowest grade in the second test?** – the goal was to see if users interacted with the bar chart that shows the distribution of grades, by clicking on the bar for the lowest grade, and then identified the student from the table below the chart.

All seven users managed to complete every task and answer the questions correctly. On task 5.4, one user mentioned that they expected that by clicking a bar in the chart that shows the distribution of grades, the bar would change colour, to indicate highlighting – this was a very pertinent suggestion and this behaviour was implemented, as can be seen in Figure 3.9. Moreover, at the time of the test, the button to add a course was located at the top-right corner of the list of courses. Two users had trouble finding the button when performing task 1. They said that they were expecting to find the button on the navigation sidebar because adding a course was a very important action – the button ended up being moved to the navigation sidebar.

The next feature that was tested was the Students feature. The tasks created for the test were the following:

1. **How many students are mentors in the academic year 2018/2019?** – the goal was to check if users read the labels on top of the students lists that showed the number of elements in the list,
2. **How many of these mentors are from the “LEIC” or “MEIC” degrees?** – the goal was to check if users could use the filter controls,
3. **How many mentors are not registered in the academic year 2018/2019?** – the goal was to check if users could access the page to view unregistered students,
4. Access the page of a specific mentee.

4.1. In which evaluation did the student obtain the lowest grade? – the goal was to check if the users could interpret the chart with the evolution of grades over time, on the Grades tab of the mentee’s page,

4.2. What grade did the student obtain in the first test of “Calculus”? – the goal was to check if users could find the grade in the tables on the Grades tab of the mentee’s page,

4.3. How many tasks did the mentor assign to the student? And how many of these were completed by the student? – the goal was to check if users could read the stacked bar chart on the Study Tools tab of the mentee’s page,

4.4. Switch the current mentor of that student for the given mentor,

5. Enrol the given mentors in the academic year of 2018/2019 – the goal was to check if users could interpret the on-screen instructions on how to create the Excel sheet to enrol the mentors.

In this test, three out of the five users failed task 4.2. Instead of using the chart that plots the evolution of grades, these users tried to answer the question by checking the tables that display the student’s grades for each course, and chose the lowest value. Their answer was wrong because the grade intervals (minGrade and maxGrade) were different, and the grade that they ended up choosing was the wrong one. After the test, these users were told their answer was wrong, and asked what their reasoning was. The users said that they did not notice the chart and did not think to interact with it. They also said that they did not think to check the grade intervals. This issue will have to be investigated and tested again in a future round of usability testing.

The third and final test concerned the Feedback and Dashboard features, and the tasks were the following:

1. How many feedback entries have the word “canoagem”? – the goal was to check if users used the search bar to filter the feedback,

2. Which recent evaluation has the most negative grades? – the goal was to check if users used the “recent evaluations” card in the Dashboard,

3. What was the day of the past week with the most activity logs related to “Tasks”? – the goal was to check if users used the filter below the charts, in the “activity logs” card in the Dashboard, to view only the logs related to “Tasks”.

In this final test, tasks 1 and 3 prompted user errors. In task 1, two out of five users did not use the search bar to look up the word, instead they checked every entry in the list, manually looking for those that had the word. After the test, when asked why they did not use the search bar, they said they did not notice it was there – this issue will also be explored in the future. A more critical error was the one that happened in task 3. Three out of five users failed this task, and the remaining two initially gave an incorrect answer (but rectified it after some time). None of the users used the filter below the chart in the “activity logs” card, and thus answered with the day of the week with the most activity logs in total, not related to “Tasks”. One of the users explained that the styling of the filter controls did not catch their attention, and also that they would expect to find the filters above the charts, not below. This is a problem that will be investigated further and tested in a future round of usability testing.

Finally, to gain a general idea of how the users felt about the system, they answered the System Usability Scale (SUS) survey, which is a means of assessing the usability of a system. The survey consists of 10 items (Figure 4.2) which the users must answer on a scale of 1 to 5 (1 corresponding
to “strongly disagree”, and 5 to “strongly agree”). The answers are then converted into a score which ranges between 0 and 100.\[21\] An acceptable average score on the System Usability Scale ranges between 65 and 70.\[22\] The users answered a version of the SUS survey translated into European Portuguese.\[23\] Table 4.1 shows the average scores obtained by the features that were tested, which fall above the range of acceptable average scores. While there is no intention to use these scores as a means to quantitatively evaluate the Web application and to make a definite statement about its usability, it is still interesting to note that the Feedback and Dashboard features have the highest score, even though this test had a task that every user failed. This could be explained by the fact that the SUS survey is answered immediately after the users finish the test, before any feedback is given about the results.

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Figure 4.2: The ten items on the System Usability Scale \[21\]

<table>
<thead>
<tr>
<th>Average SUS Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courses 95</td>
</tr>
<tr>
<td>Students 93</td>
</tr>
<tr>
<td>Feedback and Dashboard 97</td>
</tr>
<tr>
<td><strong>Average of all tests</strong> 95</td>
</tr>
</tbody>
</table>

Table 4.1: The average SUS scores obtained by the Web application
4.3 Testing NAPP with experimental and control groups

All the work documented in this thesis allowed to get NAPP to a state where it is ready to undergo a bigger and more formal test, in a real environment. The goal of this test is to evaluate whether NAPP contributes positively to the MP-TP by evaluating a key metric:

*How many days have passed since a professor published an evaluation’s grades until the coordinator received them*

For this test, the students will be divided into two groups. An experimental group will use NAPP’s mobile application and, to be able to make a comparison with the traditional method of the MP-TP, a control group of roughly the same size will not use the mobile application. This test will include students of two academic degrees: “LEIC” and “LEGI”. The “LEIC” degree has approximately 90 mentees and the “LEGI” degree has approximately 60 mentees. The groups will be defined such that:

- The number of mentees in each group should be balanced, with each group having roughly half of the mentees of each degree,

- A mentor must either use the mobile application or not, the mentor must not have mentees in both groups,

- The students in each group should be selected at random.

The mentees in the experimental group will report their grades through the mobile application. With NAPP, the reported grades are instantly made available to the coordinator, so the metric is equivalent to “how many days have passed since a professor published an evaluation’s grades until the mentee reports their grade.” Since NAPP creates a document in the Log database every time a grade is reported, we will only have to keep a record of the dates when the professors publish the grades.

In the control group, students will report their grades without NAPP. Because of the “boomerang” flow mentioned in the Introduction and pictured in Figure A.1, we will need to measure the day when the mentor sends a mentee’s grade to the coordinator, because this is effectively the day when the coordinator receives it. Using only the coordinator’s collaborative spreadsheet, which is filled in by mentors, it would be very difficult to keep a record of the dates when mentors reported each grade. However, we still wanted the coordinator to be able to analyse the mentees’ grades in a familiar way by using the same spreadsheet. The solution was to create a survey on the Google Forms platform, which mentors will use to report their mentees’ grades, by submitting a response to the survey for each grade. The platform registers the date of each submission, and ensures that the submissions are automatically added to a Google Sheets spreadsheet. By making use of the functionalities of Google Sheets, it was possible to create a programme that runs in the background, which will take the grades submitted and copy them to the coordinator’s usual spreadsheet.

During the test, the coordinator will simultaneously use NAPP’s Web application to analyse the academic performance of the mentees in the experimental group, and the usual spreadsheet to analyse the academic performance of those in the control group.

This test will take place from late-October 2018 until January 2019 (end of the first semester).
Chapter 5

Conclusions and Future Work

The unstructured beta test conducted with a small group of students over a long period revealed that students engaged with the mobile application at least once a week, except for one week. The usability test carried out on the new Web application allowed to diagnose some usability problems, but users were generally very satisfied with the application.

In the future, the focus will be on the formal test that will be conducted with a large group of students, in order to validate the hypothesis that NAPP will add value to the Mentoring Programme in Taguspark. Given the modular nature of NAPP, it could be adapted to programmes with a similar operation, such as tutoring and support of mobility students. Further enhancements should be made to NAPP, including the implementation of push notifications in the mobile application to increase user engagement, as well as the implementation of a score system of mentors’ activities by the MP-TP’s coordinator.
Bibliography


Appendix A

Academic success in the Mentoring Programme before NAPP
Figure A.1: A BPMN model of the MP-TP’s process for academic success prior to the implementation of the NAPP framework. The MP-TP’s coordinator asks the mentor for their mentee’s grades, and consequently the mentor requests that the mentee reports their grades back. The mentee then reports their grades directly to the mentor, who will then send those grades to the MP-TP’s coordinator. After receiving the grades, the coordinator analyses the them, and assesses whether the mentee needs help or not. If the mentee has no negative grades, they do not need help, and the process will terminate. However, in case of one or two negative grades, the coordinator will ask the mentor to help the mentee. In case of more than two negative grades, the coordinator will interview the mentee and provide personalised support.
Appendix B

Academic success in the Mentoring Programme with NAPP
Figure B.1: A BPMN model of the MP-TP’s process for academic success. The process starts when the mentee reports their grades using NAPP’s mobile application, and the grades are instantly available to the mentor and the MP-TP’s coordinator. Once the grades have been reported, the mentor will, at their discretion, check the mentee’s grades and check up on the mentee. Once the MP-TP’s coordinator receives the grades, the remainder of the process will be identical to that described in Figure A.1, since this part of the process did not change with the implementation of NAPP.
Appendix C

Guide for the structured test
### LEIC

<table>
<thead>
<tr>
<th>Mentor</th>
<th>Mentorando 1</th>
<th>Mentorando 2</th>
<th>Mentorando 3</th>
<th>Mentorando 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atribuir tarefa ao mentorando 1, 2 e 4: &quot;estudar para o teste de CDI-II&quot;</td>
<td>Criar tarefa: &quot;Estudar Álgebra&quot;</td>
<td>Reportar nota de 16 valores obtida no teste de MD</td>
<td>Criar tarefa: &quot;Estudar para o teste de LP&quot;</td>
</tr>
<tr>
<td></td>
<td>Verificar notas: - mentorando 2 -&gt; 16 MD; - mentorando 3 -&gt; 11 CDI-II.</td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
</tr>
<tr>
<td></td>
<td>Atribuir tarefa ao mentorando 2 e 3: &quot;reunião sexta no bar amarelo, às 16h&quot;</td>
<td>Reportar nota de 4 valores obtida no 1º teste de Álgebra</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
<td>Dar sugestão: &quot;Gostei da app do mentorado! :)&quot;)&quot;</td>
</tr>
<tr>
<td></td>
<td>Verificar notas: - mentorando 1 -&gt; 4 1º Teste Álgebra; - mentorando 3 -&gt; 1 1º Teste Álgebra.</td>
<td>Concluir tarefa: &quot;Estudar Álgebra&quot;</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
<td>Concluir tarefa: &quot;Estudar IAED&quot;</td>
</tr>
<tr>
<td></td>
<td>Abrir a app</td>
<td></td>
<td></td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
</tr>
</tbody>
</table>
|        | Verificar notas: - mentorando 1 -> 4 1º Teste Álgebra; - mentorando 3 -> 1 1º Teste Álgebra; - mentorando 4 -> 13 1º Teste LP. | Realizar 1 ciclo de Pomodoro. | Realizar 1 ciclo de Pomodoro | Realizar 1 ciclo de Pomodoro.
### Mentor LEIC

<table>
<thead>
<tr>
<th>Tarefa</th>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhã</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atribuir tarefa ao mentorando 1, 2 e 4: &quot;estudar para o teste de CDI-II&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tarde</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verificar notas: - mentorando 2 -&gt; 16 MD; - mentorando 3 -&gt; 11 CDI-II.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manhã</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atribuir tarefa ao mentorando 2 e 3: &quot;reunião sexta no bar amarelo, às 16h&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tarde</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verificar notas: - mentorando 1 -&gt; 4 1ºTeste Álgebra; - mentorando 3 -&gt; 1 1ºTeste Álgebra.</td>
<td></td>
<td></td>
<td>Abrir a app</td>
</tr>
<tr>
<td>Estado Esperado</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verificar notas: - mentorando 1 -&gt; 4 1ºTeste Álgebra; - mentorando 3 -&gt; 1 1ºTeste Álgebra; - mentorando 4 -&gt; 13 1ºTeste LP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dia 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dia 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dia 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Igual ao estado esperado?                                              |       |       |       |
| Online                                                                |       |       |       |
| Offline                                                               |       |       |       |

**Mentorando 1:** __________________________________________

**Mentorando 2:** __________________________________________

**Mentorando 3:** __________________________________________

**Mentorando 4:** __________________________________________
### 1 Mentorando LEIC

<table>
<thead>
<tr>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manhã</strong></td>
<td><strong>Manhã</strong></td>
<td><strong>Manhã</strong></td>
</tr>
<tr>
<td><strong>Tarefa</strong></td>
<td><strong>Tarefa</strong></td>
<td><strong>Tarefa</strong></td>
</tr>
<tr>
<td>Criar tarefa: &quot;Estudar Álgebra&quot;</td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Reportar nota de 4 valores obtida no 1º teste de Álgebra</td>
</tr>
<tr>
<td>Estado Esperado</td>
<td>Aparece uma nova tarefa: &quot;estudar para o teste de CDI-II&quot;. Esta deve ser movida para a lista de tarefas concluídas</td>
<td>Aparece uma entrada para a nota na lista de notas reportadas</td>
</tr>
<tr>
<td>Igual ao estado esperado?</td>
<td>Tarefa é movida para a lista de tarefas concluídas</td>
<td>-</td>
</tr>
</tbody>
</table>

### 2 Mentorando LEIC

<table>
<thead>
<tr>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manhã</strong></td>
<td><strong>Manhã</strong></td>
<td><strong>Manhã</strong></td>
</tr>
<tr>
<td><strong>Tarefa</strong></td>
<td><strong>Tarefa</strong></td>
<td><strong>Tarefa</strong></td>
</tr>
<tr>
<td>Reportar nota de 16 valores obtida no teste de MD</td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Criar tarefa: &quot;Estudar IAED&quot;</td>
</tr>
<tr>
<td>Estado Esperado</td>
<td>Aparece uma nova tarefa: &quot;estudar para o teste de CDI-II&quot;. Esta deve ser movida para a lista de tarefas concluídas</td>
<td>Nova tarefa criada</td>
</tr>
<tr>
<td>Igual ao estado esperado?</td>
<td>Nenhuma tarefa nova</td>
<td>Tarefa é movida para a lista de tarefas concluídas</td>
</tr>
</tbody>
</table>
### 3 Mentorando LEIC

<table>
<thead>
<tr>
<th>Dia</th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tarefa</strong></td>
<td>Reportar nota de 11 valores obtida no 1º teste de CDI-II</td>
<td>Dar sugestão: &quot;Não me identifico com o programa Mentorado&quot;</td>
<td>Reportar a nota de 1 valor obtida no 1º teste de Álgebra</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
<td>Realizar 1 ciclo de Pomodoro</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
</tr>
<tr>
<td><strong>Estado Esperado</strong></td>
<td>Aparece uma entrada para a nota na lista de notas reportadas</td>
<td>-</td>
<td>Aparece a nota reportada</td>
<td>Nenhuma tarefa nova</td>
<td>-</td>
<td>Aparece uma nova tarefa: &quot;reunião sexta no bar amarelo, às 16h&quot;</td>
</tr>
</tbody>
</table>

**Igual ao estado esperado?**

---

### 4 Mentorando LEIC

<table>
<thead>
<tr>
<th>Dia</th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tarefa</strong></td>
<td>Criar tarefa: &quot;Estudar para o teste de LP&quot;</td>
<td>Verificar se o mentor me atribuíu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Dar sugestão: &quot;Interface fácil de utilizar&quot;</td>
<td>Concluir tarefa: &quot;Estudar para o teste de LP&quot;</td>
<td>Reportar a nota de 13 valores obtida no 1º teste de LP</td>
<td>Realizar 1 ciclo de Pomodoro</td>
</tr>
</tbody>
</table>

**Estado Esperado**

Nova tarefa criada

**Igual ao estado esperado?**

---
<table>
<thead>
<tr>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mentor</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atribuir tarefa ao mentorando 2: &quot;estudar para o teste de MO&quot;</td>
<td>Verificar nota: - mentorando 2 -&gt; 7 Microeconomia.</td>
<td>Atribuir tarefa ao mentorando 2: &quot;reunião segunda no bar azul, às 12h&quot;</td>
</tr>
<tr>
<td></td>
<td>Verificar nota: - mentorando 1 -&gt; 14 1ºTeste MO.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Abrir a app</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verificar notas: - mentorando 1 -&gt; 10 1ºTeste Álgebra, - mentorando 1 -&gt; 14 1ºTeste MO.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Online</strong></th>
<th><strong>Offline</strong></th>
<th><strong>Online</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mentorando 1</strong></td>
<td>Criar tarefa: &quot;Estudar Ciência de Materiais&quot;</td>
<td>1ºTeste MO.</td>
</tr>
<tr>
<td></td>
<td>Dar sugestão: &quot;App fácil de utilizar&quot;</td>
<td>Realizar 1 ciclo de Pomodoro</td>
</tr>
<tr>
<td></td>
<td>Reportar nota de 14 valores obtida no 1º teste de MO</td>
<td>Reportar nota de 10 valores obtida no 1º teste de Álgebra</td>
</tr>
<tr>
<td></td>
<td>Concluir tarefa: &quot;Estudar Ciência de Materiais&quot;</td>
<td>Verificar se o mentor me atribuiu alguma tarefa em caso afirmativo devo concluí-la</td>
</tr>
<tr>
<td><strong>Mentorando 2</strong></td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
</tr>
<tr>
<td>Reportar nota de 7 valores obtida no teste de Microeconomia</td>
<td>Criar tarefa: &quot;Estudar CDI-II&quot;</td>
<td>Concluir tarefa: &quot;Estudar CDI-II&quot;</td>
</tr>
<tr>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
</tr>
</tbody>
</table>

**LEGI**
## Mentor LEGI

<table>
<thead>
<tr>
<th>Tarefa</th>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atribuir tarefa ao mentorando 2: &quot;estudar para o teste de MO&quot;</td>
<td>Online</td>
<td>Atribuir tarefa ao mentorando 2: &quot;reunião segunda no bar azul, às 12h&quot;</td>
<td>Abrir a app</td>
</tr>
<tr>
<td>Verificar nota: - mentorando 2 -&gt; 7 Microeconomia.</td>
<td>Offline</td>
<td>Verificar nota: - mentorando 1 -&gt; 14 1ºTeste MO.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estado Esperado</th>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aparece a nota do mentorando</td>
<td>Online</td>
<td>A nota não aparece</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Igual ao estado esperado?</th>
<th>Dia 1</th>
<th>Dia 2</th>
<th>Dia 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Online</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mentorando 1:** ___________________________________

**Mentorando 2:** ___________________________________
### 1 Mentorando LEGI

<table>
<thead>
<tr>
<th></th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tarefa</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Criar tarefa: &quot;Estudar Ciência de Materiais&quot;</td>
<td></td>
<td>Dar sugestão: &quot;App fácil de utilizar&quot;</td>
<td>Reportar nota de 14 valores obtida no 1º teste de MO</td>
</tr>
<tr>
<td><strong>Estado Esperado</strong></td>
<td>Nova tarefa criada</td>
<td>-</td>
<td>Aparece uma entrada para a nota na lista de notas reportadas</td>
<td>-</td>
<td>Aparece uma entrada para a nota na lista de notas reportadas</td>
<td>Tarefa é movida para a lista de tarefas concluídas</td>
</tr>
<tr>
<td><strong>Igual ao estado esperado?</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2 Mentorando LEGI

<table>
<thead>
<tr>
<th></th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
<th>Manhã</th>
<th>Tarde</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tarefa</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reportar nota de 7 valores obtida no teste de Microeconomia</td>
<td>Verificar se o mentor me atribuiu alguma tarefa e em caso afirmativo devo concluí-la</td>
<td>Criar tarefa: &quot;Estudar CDI-II&quot;</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
<td>Concluir tarefa: &quot;Estudar CDI-II&quot;</td>
<td>Verificar se o mentor me atribuiu alguma tarefa</td>
</tr>
<tr>
<td><strong>Estado Esperado</strong></td>
<td>Aparece uma entrada para a nota na lista de notas reportadas</td>
<td>Aparece uma nova tarefa: &quot;estudar para o teste de MO&quot;. Esta deve ser movida para a lista de tarefas concluídas</td>
<td>Nova tarefa criada</td>
<td>Nenhuma tarefa nova</td>
<td>Tarefa é movida para a lista de tarefas concluídas</td>
<td>Aparece uma nova tarefa: &quot;reunião segunda no bar azul, às 12h&quot;</td>
</tr>
<tr>
<td><strong>Igual ao estado esperado?</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>