Educado

An educational mobile application for Brazillian waste pickers

Authors
Beniamin Lobodziec
Hijab Nasir
Jonathan Eis Benzon
Karl-Emil Hertz
Lojain Ajek
Nicolai Uhre Mandrup
Tommy Grenaae

AALBORG UNIVERSITET

Aalborg University
Electronics and IT

Department of Computer Science
Aalborg University
http://www.aau.dk

Title:
Educado - An educational mobile application for Brazillian waste pickers

Theme:
Complex Front-end Developing

Project period:
5. SEMESTER

Project group:
Group 3

Group member:
Beniamin Lobodziec
Hijab Nasir
Jonathan Eis Benzon
Karl-Emil Hertz
Lojain Ajek
Nicolai Uhre Mandrup
Tommy Grenaae

Supervisor:
Andres R. Masegosa

Number of pages: 69

Due date:
December 20, 2022

Abstract:

This report covers the process of a 5th-semester Software Engineering Bachelor group working on a large-scale project in collaboration with other groups of the same semester. The report focuses on documenting how further development on the front-end of the educational app *Educado* evolved throughout the semester with the use of agile software development techniques. The report consists of six parts, where each increment builds on top of the previous one. The main focus of these increments was the process of how the different teams collaborated to develop the same product. The initial starting point of this product was developed by two Bachelor students during their Bachelor project. During this semester, 20 students took over and have been working on different aspects of the *Educado* product. The goal for this project is to have students develop the first iteration which can then be an open-source project that future students will be able to take over. We used field research from the 2019 Smart Caching Learning Materials done by the same students who made the initial product. The increments of the front-end implementations creates a new foundation for future development.

Preface

Credits:

We would like to thank our supervisor Andres R. Masegosa for helping us create a clearer path towards the end goal of the project, and for being able to supervise with little to no agenda from the group.

Another mention also goes out to our semester coordinator Daniel Russo, who participated in sprint reviews throughout the project of his own free will, for us to learn how to improve in our process.

Furthermore, we would like to thank Jacob Vejlin Jensen and Daniel Britze who made the initial product, and acted as stakeholders during the project.

Finally, a big thanks go to Mateus Halbe Torres, who acted as Product Owner for the development of the Educado-project.

Resources:

The GitLab repository: `https://gitlab.com/A-tech/educado-mobile-application`
Currently the repository is closed, but the plan is to open it up.
Contact Daniel Britze at daniel@somethingnew.dk if you need help with access.

Contents

Part I

Introduction

1 Background

Written in collaboration with all groups

One of the world's largest landfills lies in Brazil [1]. The landfill was closed in 2018 which is a positive development in regard to the current climate crisis and the general health of the people working and living in or near the landfill [2].
When the landfill was closed, the waste pickers who lived and worked there were instead provided with jobs at the newly created recycling centres to sort the waste. This has led to a lower income for the waste pickers in general, even though they were hired at the recycling facilities. Furthermore, the payment went from a daily to a weekly salary, which meant that the waste pickers had a harder time managing the money that was available to them. The waster pickers do not have many other options of employment, as many of them do not have any education, which could lead them to other more profitable jobs [3, p. 3].

This problem identification has led to the creation of the *Educado* project, a digital learning platform for waste pickers in Brazil. It began in 2019 when students at Aalborg University collaborated with students at the University of Brasilia to create a mobile application that can be used to educate waste pickers.
In order to complete this, an application was created that apart from providing basic education also gives certifications for completed courses. This is meant to help these people improve their standards of living. The application should be intuitive and contain audio and video clips instead of text when possible, to make it more available for people who are illiterate.
A web application was also created, allowing 'Content Creators' to build, modify and publish courses all from a web browser [4].

This semester, three groups at Aalborg University are collaborating to continue this project and its vision. The project's goal is to clean up and refactor the existing code and database, as well as redesign the mobile- and web applications and add new functionality. As we are working on a shared project some sections in the report have also been written in collaboration between two or more groups. These sections' beginning and end will be clearly marked throughout the report.

1.1 Important concepts

Throughout the report, there will be used terminology from Scrum and agile software development, so it would come to good use if the reader has a basic knowledge of the different kinds of software development methods. In this project, we are working as members of a development team following (while learning) Agile practices. This section briefly describes key concepts of the Agile framework called Scrum that we are meant to work within. For in-depth explanations of these concepts, explore original resources on "*The Agile Manifesto*" [5] and Scrum [6].

General Concepts

The following concepts are general for Scrum.

- The **Product Backlog** is a list of all the items or work that needs to be done to achieve the desired state of the product.
- A **Sprint Backlog** is a subset of the Product Backlog containing the work required to fulfil a sprint's goal.
- An **Increment** represents the finished valuable work the Developers have completed during a sprint.
- A **Definition of Done** formally defines a baseline for the state of quality an Increment is required to fulfil for the product. It is about the activities needed to ensure a certain level of quality of the work being done on the backlog items before they can be considered an Increment.

- The **User Story** is not described in the Scrum Glossary because it is not a mandatory part of Scrum [7]. However, it is perfectly compatible with Scrum, the user story aims to express requirements from a user's perspective. It is what the user needs from the system. In Scrum, it is the PO's responsibility to relay this information to the developers [8]. In this project, we, the developers, have made the user stories based on our understanding of the system and then confirmed them in meetings with the PO.
- The concept of **Scaling** becomes relevant when a developer team or software product grows in size and complexity, possibly becoming unmanageable. A scaling framework attempts to set up the rules or structure for managing these challenges. One such framework, Nexus [9], minimally builds upon Scrum to enable three to nine teams to work together on a single product.

Roles in Scrum

Key people and their roles in Scrum are listed below:

- The **Scrum Master** is the head of the Scrum team. First, among his peers of developers, he is responsible for the effectiveness of the team.
- The **Product Owner** (PO) ensures that the team produces a valuable product by being the link between developers and stakeholders.
- The **Developers** are the members of the scrum team creating the product.
- A **Scrum Team** consists of a Scrum Master, a PO, and the Developers.
- The **Stakeholders** are people outside of the scrum team, who have knowledge of or an interest in the product. They are represented by the PO and should play an active role at Sprint Review.

Events in Scrum

Defining activities in Scrum that have been central to this project:

- The **Sprint** is a time-restricted period during which all the other events take place.
- **Sprint Planning** marks the start of a sprint, where the scrum team chooses from the Product Backlog which task is the most valuable to work on during the coming sprint.
- **Daily Scrum** is a daily 15-minute event to inspect previous work and lay out the work plans for the following day.
- At **Sprint Review** the Increment is inspected by the Scrum Team and Stakeholders, its value assessed, and the Product Backlog updated.
- During **Sprint Retrospective** the Scrum Team evaluates the ending sprint and tries to find ways to improve future sprints.

1.2 Rich Picture

The previous developers of this project made a rich picture which can be seen below.

It showcases how the waste pickers are encouraged by the government to make use of these proposed sorting facilities, to help educate them on key areas for their well-being such as sanity, economy, and more. Hereby, we see a draft for a solution to the problem, that is the employee at the sorting facility hired by the government will be able to guide and educate the waste pickers via an application on their phones. By using the application, the employee of the sorting facility can distribute content to the waste pickers that are relevant to them.

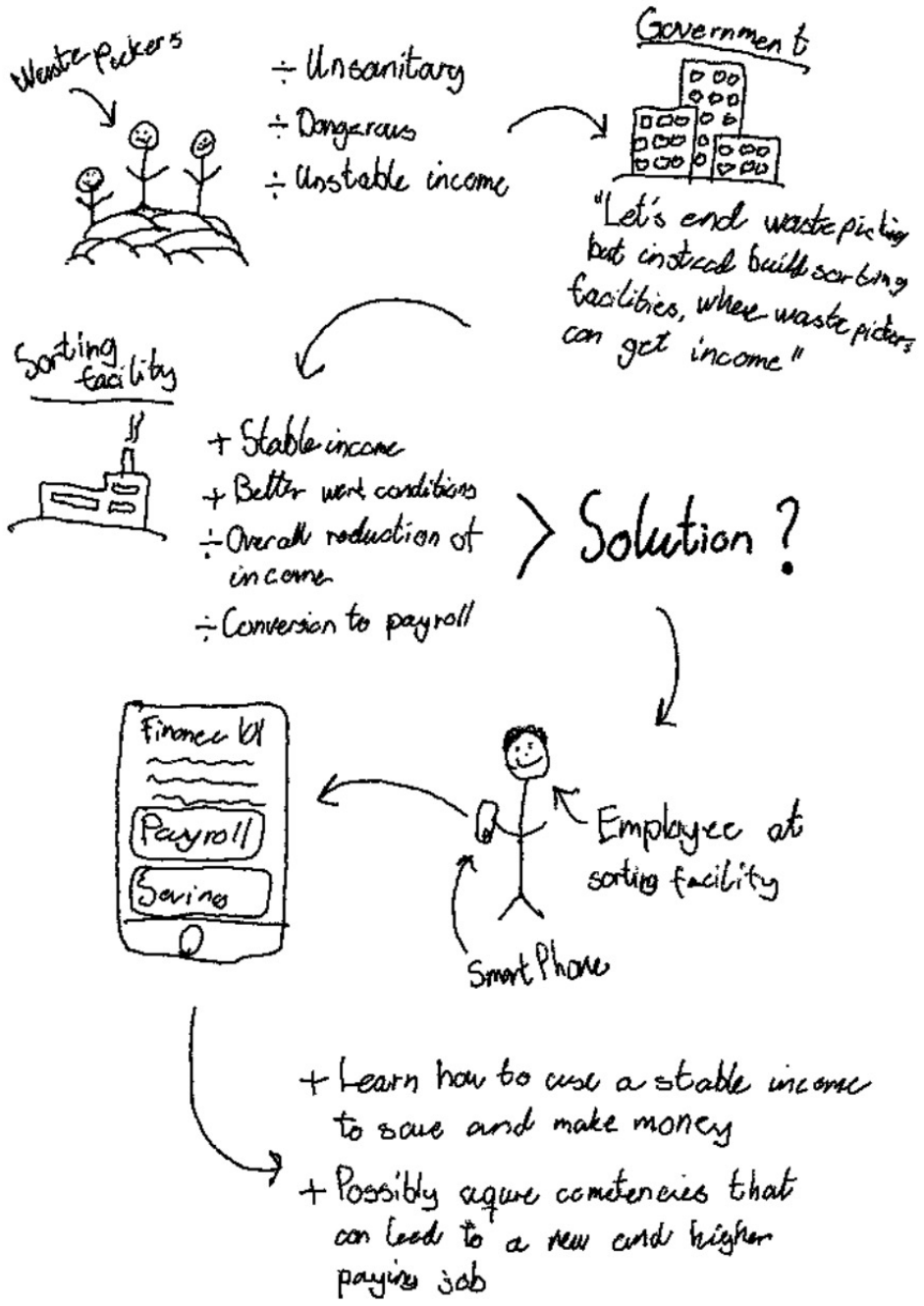


Figure 1.1: Rich Picture from the previous project group

1.3 Initial state of the Educado project

Currently, there are three major components of the project. The web application, the back-end, and the mobile application. There are three different users: The administrator, the content creator, and the waste picker that is in the need of learning. The web application is a gateway for the administrators and the content creators, while the mobile application is the gateway for the waste pickers. The back-end is communicated with both the web and mobile front-ends. The administrators (and content creators) create the courses for the platform on the web application and the Brazilian waste pickers attend the courses through the mobile application. The administrator is the one controlling the *Educado* Mobile Learning Platform.

Web application

When the content creator logs into the web application, they do it through Google's authentication system, which requires a Google account. So far, this is the only form of authentication implemented on the platform. Thus, storing the user data has not been dealt with. After logging in, the content creator is presented with their courses, presented as a square tile with a title and an image (if uploaded). For each course, there is a button with a title that says "edit". Furthermore, on the initial page, the employee has a button to log out in the upper right corner. There are two buttons on the left side. One that takes you to your courses and one that lets you create a new course.

The Web Application has a somewhat simple design so far, as not a significant amount of time has been put into it.

When creating a new course, you can add a title, add a description, choose a category (such as *Health*, *Finance*, *Technology* and *Language*) and upload a cover image. Each course is divided into sections and each section can be further divided into different components, such as text, images, videos, or audio clips. Each of these components can be added as desired. In total, these elements make up what is essential for a full-fledged course. Furthermore, each section can be moved around, such that the course creator can structure their course, down to each element. Fundamentally, a course is made up of n number of sections, and each section consists of m number of components, that are either text, videos, images, or audio.

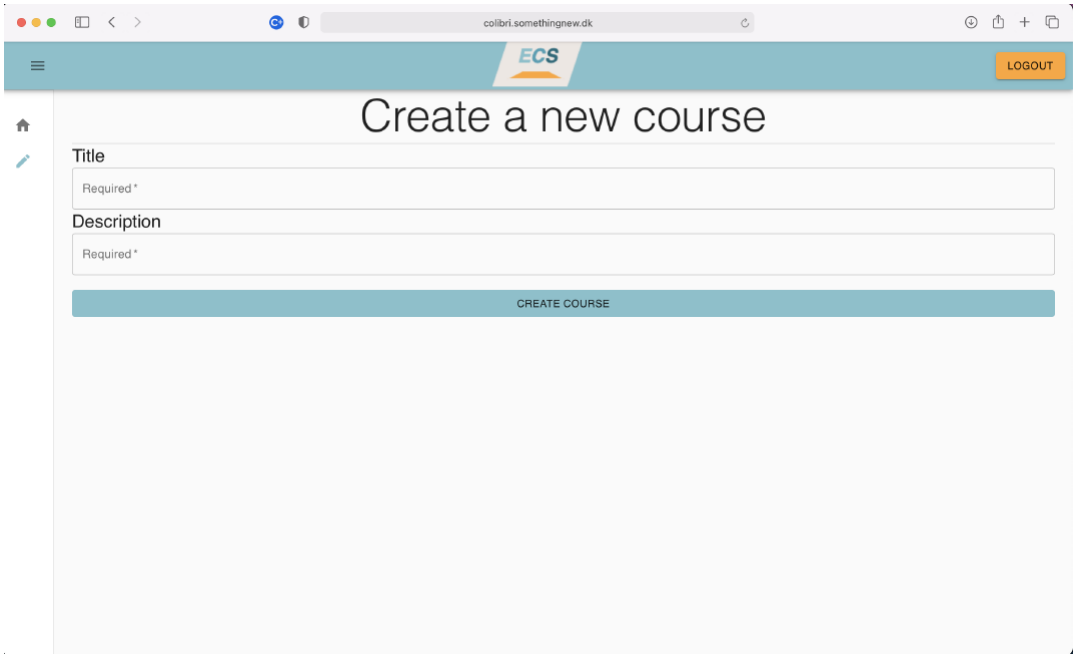


Figure 1.2: Previous state of the web application's front-end

Mobile Application

Currently, the mobile application is a simple React Native application that gives you access to courses. When launching the mobile application, you are presented with a screen that lists all the courses you are actively participating in. Currently, all elements are presented in a similar way as the web application.
Upon opening the mobile application, we encounter the following main tab as seen in Figure 1.3, where the user is able to see the active courses that he/she is currently following, and a status over their progress. There is a demo course available where the user can see what a course looks like so they get more familiar with the functionalities of the app.

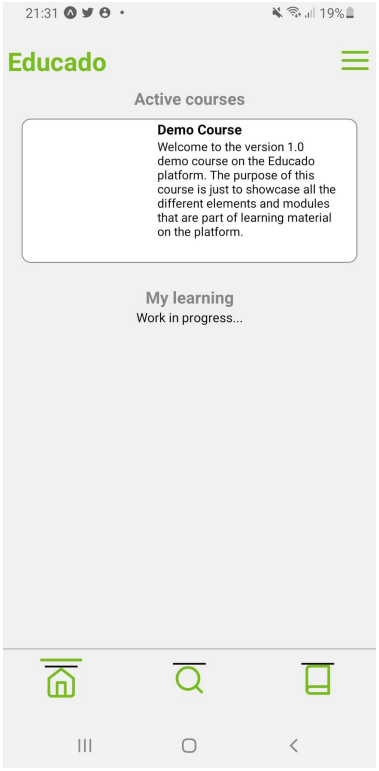


Figure 1.3: Main Tab

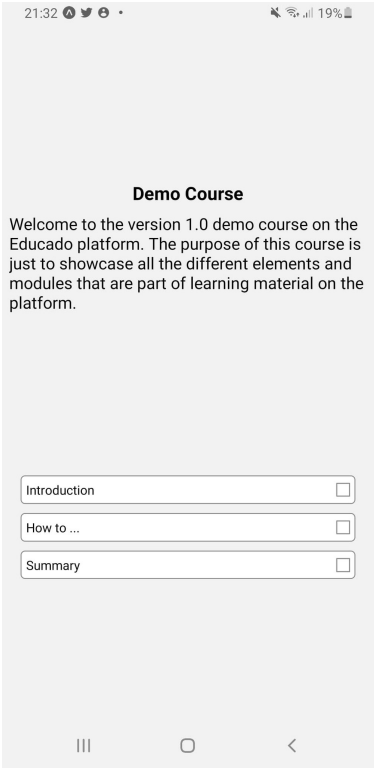


Figure 1.4: Course Tab

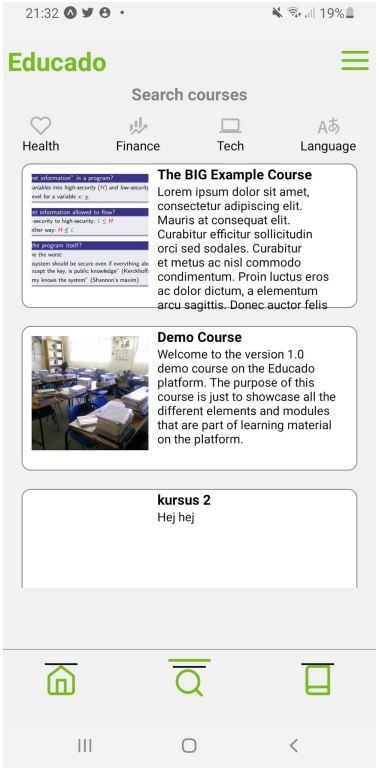


Figure 1.5: Search Tab

Since we know that the users of the mobile application are people with minor educational backgrounds, it is important to limit the amount of text in order to make it easier for the users to interact with the application. This is not the case with this initial solution as it contains a lot of text, which can create trouble for the user experience and in worst case end up with them not using the application. Besides that, there is a button at the top-right corner where it supposedly should bring down a menu to navigate to different parts of the application, but it has not yet been implemented. Moreover, there are three buttons at the bottom:

- Course screen
- Search
- Library

By clicking on the Demo Course shown in Figure 1.3: Main Tab, we can see an example of how a course would look.

In Figure 1.4: Course Tab, the user can see the introduction to the course, learn how to complete and then get a summary of the results.
By clicking on the Back button the user returns to the main screen. From there if the user clicks on the Search-button, the screen from Figure 1.5: Search Tab, will show up.
On this page, it is possible for users to browse between available courses from different categories.

1.4 Class Diagram of the Data Model

This section will give an overview of which classes were featured in the project when the project was handed over. In this project, we will try and expand upon it and may change a couple of details, which will be introduced later. In the project's source code, all three classes have additional variables that are never used in code and thus are not represented in the class diagram.

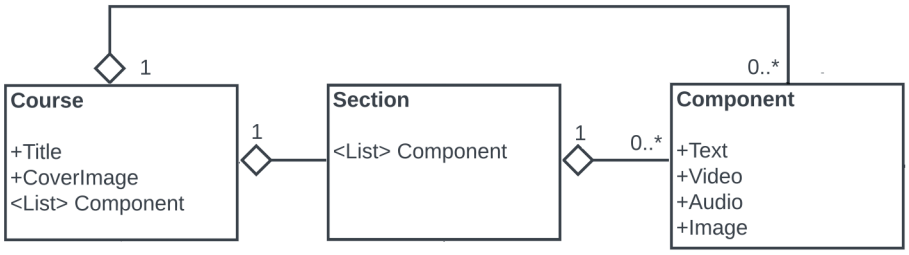


Figure 1.6: Class diagram of the current Educado mobile application

Classes

- **Course**

  Course is a class that represents a course.

- **Section**

  Section is a class that represents a section within a course.

- **Component**

  Component is a class that represents a component in a section, this can be audio, video, images, or text.

Events

The events below give us an overview of the different functionalities that are implemented by the previous project group. There will be an updated event table later in the report with events we have implemented. It shows how different classes are involved in different events, this way we may consider how the classes are related to each other and what they are able to do.

- **Open application on Android Smartphone**

  This is an event where the waste picker is able to open the application on their Android smartphone. As of now, Expo Go can be used to simulate the application

- **See list of available courses**

  This is an event where the waste picker is able to see a list of courses in that they can participate.

- **See overview of course content**

  This is an event where the waste picker is able to see all the sections of a course.

- **Load video, audio, images, and text**

  This is an event where the waste picker is able to load the content that is located in the sections in the courses.

Events	User	Course	Section
Open application on Android Smartphone	+	+	
See list of available courses	*	*	*
See overview of content	*	*	*
Load video, audio, images and text	*	*	*

Table 1.1: Event Table: “+” Indicates 0 - 1 time. “*” indicates 0 - several times

End of collaboration

2 | Prerequisites

2.1 Limitations

Written in collaboration with all groups

In this section, we describe the limitations that dictate the project's scope. The project description has set a baseline for the project limitations:

> *Improve the Educado platform and transform it into a great functional Mobile Education solution for Waste Pickers to be tested in the field by the end of the semester in Brazil.*

Besides the main goal, several high-level limitations were given to us at the start of the project.

- The waste pickers would have limited access to WiFi, which means that the application should work offline.
- The waste pickers have low to no educational background, so the application should minimize the amount of text.
- The waste pickers use older-generation mobile phones. Therefore, they have limited storage space on their phone, meaning that the application should not fill their entire storage with content.

Working from an existing project

Since this project is based on an existing solution, we are limited to working with the same underlying technologies as the original solution. This means that all further implementation will be built with:

- MongoDB
- Express (Node)
- React
- React Native

The microservice architecture from the original report also represents how the different technologies were implemented in the received project:

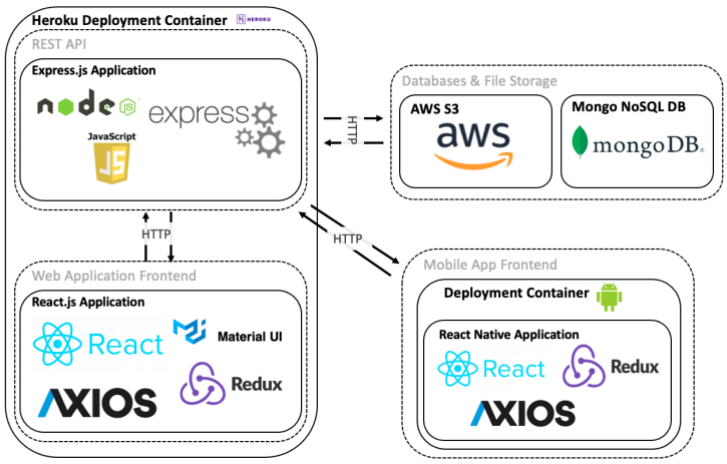


Figure 2.1: Microservice architecture from the Educado Bachelor project [4, p. 47]

The only differentiation in the technology stack that has been made, is adding type support for the React web application with Typescript, as well as some minor changes concerning the styling of the user interface in the mobile application.

Continuous learning

Our course on agile software development ran concurrently with our project, so some sprints will introduce agile development methods that do not appear in previous sprints. As such this project involved a steep learning curve regarding many new concepts and methods of agile development for multiple groups. The semester coordinator tried to help us figure out how this was done, although the approach was that we had to fail in order to learn, which caused a lot of frustration.

Furthermore, the PO was new to the role and thus had to learn what it entailed, while the project was ongoing. This meant that our own process was affected because the PO was not really clear about what was needed in the first several sprints. Furthermore, the tech stack included tech that was not familiar to the groups and thus had to be learned as the project progressed.

Deployment

Since the project is already an existing working platform, we are not going to work with the deployment of the system, and as such, further discussion about the current deployment choices will not appear in the report.

Dependencies

As described in the project description, the entire class of Software five is going to work on the same project during this semester. This means that the improvement of the existing platform is divided between the three teams, where each group is going to focus on a specific domain of the project. Group 1 will work on both front-end and back-end of the mobile application and assist with creating a link of information flow between the other two groups' implementations. Group two will focus on both back-end and front-end for the web application, for content creators who want to create educational courses for the waste pickers in Brasilia. Group three will work on creating an engaging mobile application for the waste pickers, and will primarily be in charge of the front-end side of the App.

Therefore, some of our main concerns in this project are to balance the work between fulfilling stakeholder wishes and resolving cross-team dependencies. This means that having a functionally aligned platform in every team is more important than a single group providing new features for a single aspect of the platform.

End of collaboration

2.2 Gamification

In this section, a definition of gamification will be provided. Afterwards, vital aspects of gamification will be presented, which will be tied to the different screens that were produced for the mobile application.

Gamification can be defined as "*Applying game elements and features to non-gaming software to improve user experience and participation...*" [10].

The gamified elements that are usually implemented in mobile applications are concepts like a basic score point system, badges for the completion of a task or achievement in the application, and a leaderboard. These have a positive effect on the users in general, which is why the group will use more focus on integrating these elements in the Educado application [11].

The concept of the Educado application is by nature a non-gaming software since the core functionality is to enable content creators to produce learning material for the Brazilian waste pickers to consume. From the very beginning, the PO wanted to have an engaging application with several gamification features, which meant that a lot of thought went into these when designing the different screens.

While developing, the group came up with a possible structure of where and how to use points denoted as the number of stars, badges, and leaderboard in a meaningful way. The points would be used to track the user's progression within a specific course, while the badges would be given if achievements in the application were completed. The leaderboard would serve as the main way of interconnecting related users by comparing the number of points and badges between the target user and their friends, as connecting users via a friend list proves to enhance the retention of an application: "*...the more frequent the interactions between users, the more significant the impact of social interaction on users' [...] application use*" [12].

Therefore, the group derived which gamification elements should be implemented, those are listed below:

- Points (stars)
- Badges
- Leaderboard
- Friends list

Although the group spent time thinking about how to implement all of the bullet points above, only a subset of gamification that relates more to the feel and engagement of the application, was in the end implemented, this is discussed in section (12.1)

2.3 Reading2Learn (R2L)

Reading2learn is a learning method to improve people's reading skills. This concept will be used as a baseline for the implementation of the exercise screen, where short videos will make up a vital part of the learning input. This has been done to make sure that user retention stays high and that no information in the video is lost due to attention span. This will be described further in Sprint #1 (see figure 5.4).

Reading2Learn is divided into four strategic parts which will help students to read through the curriculum easier.

1. Reading preparation

This learning part begins before the students read the text and it is where the teacher helps with grasping the essence of the text beforehand. There are two steps to this approach: Reading the text out loud for the students instead of making them read it on their own and giving the students a piece of background knowledge about the upcoming text, so it is easier to grasp without having to read themselves.

After reading a text out loud in front of the entire class, it can be necessary that the teacher also addresses important points from the text, therefore giving them background knowledge afterwards. This knowledge could also be covered in earlier sessions [13]. The concept of reading out loud the text that is found in the application will be an important feature and will be explored in the coming sprints.

2. In-depth reading

In this part of Reading2Learn, the teacher helps the students read the material independently and locate important information. One paragraph is read at a time if the text is long or challenging. The instructor directs the class to highlight two or three key ideas in each paragraph. This paragraph-by-paragraph, although it takes time, is a very efficient method for assisting all students in learning how to comprehend what they read. This part will not be implemented in the application, as it does not have that large amount of text.

3. Making notes

In this step, the students are taking notes based on the material they have read. The purpose is to highlight information in the text, so other students afterwards can instruct each other on how to write and spell different words. From an educational point of view, this is a feature that could be implemented in the application but will not be included in this version of Educado. If later down the line, the focus of the application changes to not only provide content but to also enable the waste pickers to learn how to read, this would make a lot of sense to implement.

4. Joint construction

At the end of the Reading2Learn is the joint construction step. Here, the students have to use their notes from the previous steps to create a new text in unity. The instructor will help the students take turns writing a statement for the new text, which will help them be able to fathom the entirety of the text. As the current state of the application do not revolve around producing content, this will not be used for the implementation of the application.

Part II

Sprints

3 Introduction to sprints

In this section, there will first be introduced a combined diagram of all three groups' different user stories throughout the semester (see section 3.1), after which the group's Definition of Done (see section 3.2) will be provided.

This part of the report will include the six different sprints during the project period. The general structure of each of the sprints will revolve around first introducing the user stories for the sprint, as well as how the sprint was managed. Afterwards, the focus will turn towards what cross-team communication happened, as well as what were the increments of the sprint. Lastly, every sprint will have sections revolving around which user stories were accepted or rejected at sprint review, as well as a sprint retrospective where the team reflected on the sprints workflow and discussed how to optimize it.

3.1 Combined User Stories Diagram

Written in collaboration with all groups

In an effort to better understand what happened during each sprint in the different groups. The following diagram was made to give the reader a better understanding of what has happened in the project as a whole. Figure 3.1 gives an overview of the user stories that have been completed during the six sprints. The blue 'Team COWS' have been contributing to the front-end app. The green 'Team Half Full Stack' has been contributing to both the front application and back-end capabilities inside the app, while also handling the back-end specific for the app. The yellow 'Team Sharp Deluxe' main focus is content creation and they have worked on both back-end and front-end capabilities.

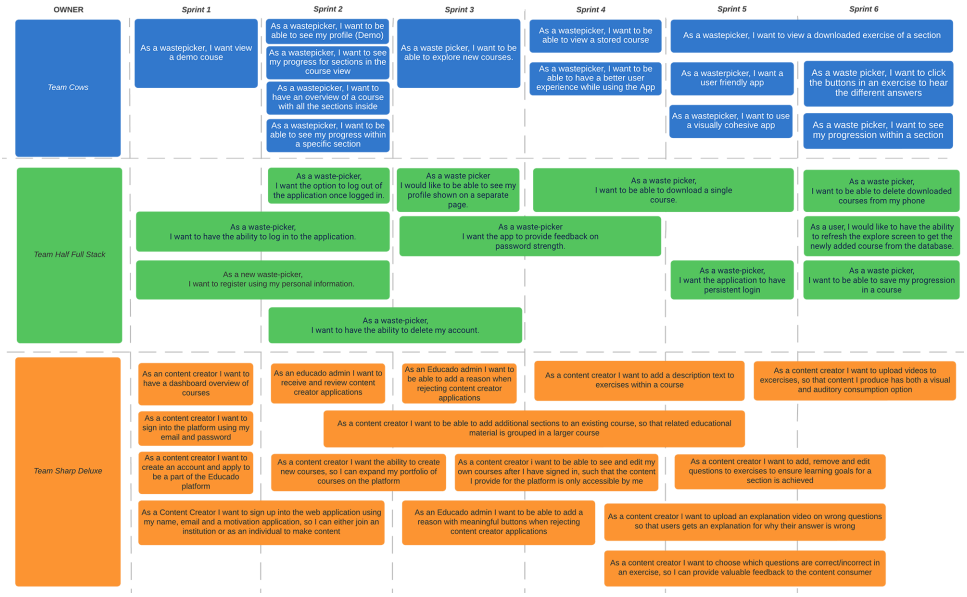


Figure 3.1: Combined User stories diagram

End of collaboration

3.2 Definition of Done

In the following, the group's Definition of Done will be presented. This is a vital part of agile software development, as it serves as a baseline for every increment done throughout the project. Reflecting on the different sprints, sprint goals, user stories, and tasks, patterns emerge that point toward what could be considered as the team's Definition of Done.

These are some of the more apparent processes that were met while developing. Thus, the following criteria had to be met for every increment to be accepted.

- A task is done when the integration test shows no sign of failure
- A task is implemented only once merged with the develop branch.
- A user story is completed once all the derived tasks are completed
- A new screen/view should go through prototyping and the design should be approved by the PO before starting implementation.
- All features implemented should comply with the PO's approval.

Note that this list was reverse-engineered throughout the sprints, as the group did not state a Definition of Done before the project started. Furthermore, it should be noted that a similar approach could be found across the other teams, which will be further discussed later in the report (see section 12).

4 System Architecture

Written in collaboration with all groups

4.1 General overview

This section focuses on the system's overall architecture which will be handed over and used for the next 5th semester's project. The ISO/IEC/IEEE 42010:2011 standards define architecture as follows:

> "*(system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution* [ISO/IEC/IEEE 42010:2011(E), p. 2]"

Architecture diagrams were used to properly represent the architecture of the system. They provide a visual representation of how the system works and which components are interacting with each other.

A representation of a high-level architecture diagram can be seen in figure 4.1. The final three repositories of the Educado system are all represented and the components use HTTP protocols to communicate over different repositories. All communication with the MongoDB database happens in the content platform, but they send and receive this information to/from the dedicated web and mobile repositories.

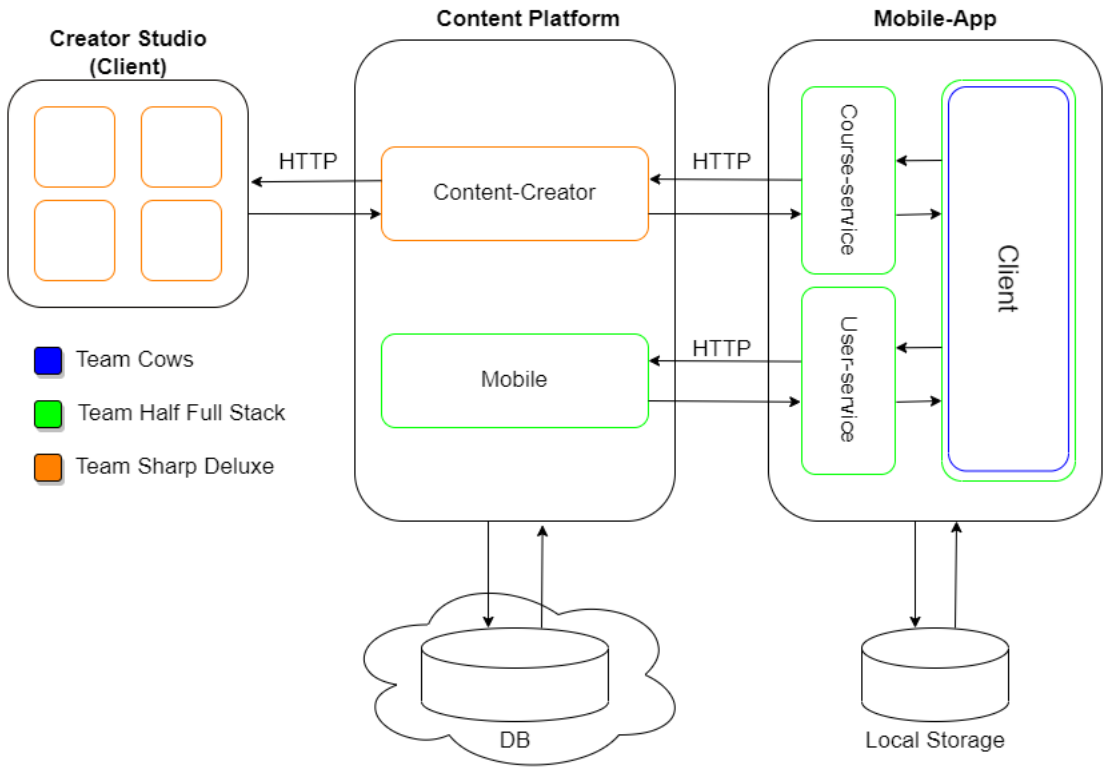


Figure 4.1: A high-level architecture diagram of the Educado system

4.2 Mobile app overview

The following is a more detailed diagram of how the mobile side of Educado is structured.

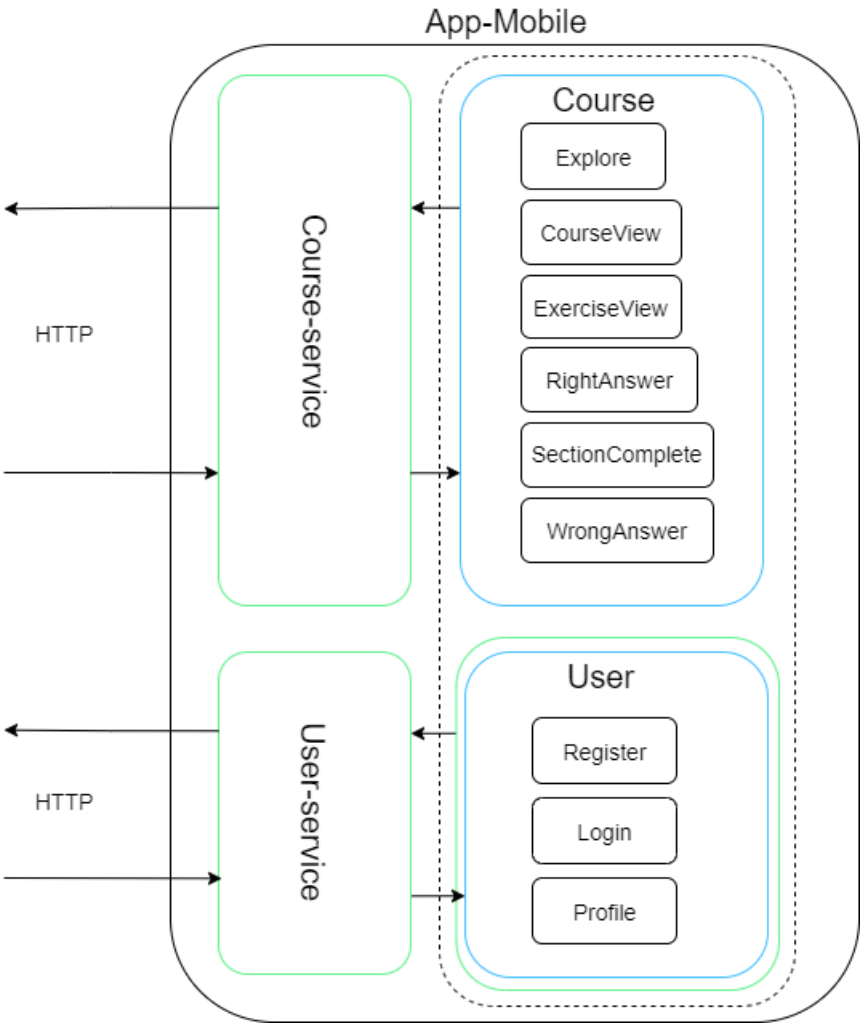


Figure 4.2: Mobile app overview

Here, the Course- and User-related aspects are coupled together in conceptual frames to better create an easy-to-understand overview. The Course part is everything that is related to the courses themselves, that be the structure, sections, exercises and content, whereas the User part relates more to the Brazilian waste picker and his/her use of the application. Internally, both frames are connected to the back-end through a service layer that handles the requests made by the user.

4.3 Back-end overview

The web application back-end is made up of a few components that each have their own set of responsibilities within the system. Each of these components is subdivided into several layers. These layers help split up the concerns by focusing on a single aspect of the application. The following is a short description of the purpose of each layer.

- **Routes**: The routing layer is responsible for the outermost interaction with the web and therefore contains all the valid endpoints of the API.
- **Controllers**: The controllers are responsible for validating the data of incoming requests and passing that data to the corresponding use cases or services.
- **Use cases**: The use cases have the responsibility of carrying out the actual steps in an operation. For example, a use case might be `approveMotivation(motivationId)` in the case of successfully signing up as a new content creator. The use case should find the existing motivation record with the given id, then update the status to approved, and finally, send an email containing a one-time password to that applicant. The use case would contain the for carrying out the steps.
- **Domain**: The domain layer is where the main entities in the system reside. For the case of content creation, examples of an entity might be the Course, Section, Category, and so on. The entities are themselves responsible for doing the state changes that always ensure that they are in a valid state. For example, an applicant's motivation should not be able to go from an accepted state to being a rejected state as this would rarely be the case in the real world. The motivation should also always include the first name and last name of the person applying. These types of rules would be enforced inside the motivation entity itself.
- **Gateways**: The gateways have the responsibility of transforming and persisting data to the database while providing a simple interface to client code.

The below diagram shows how the components of the back-end each store these layers as well as the direction of dependencies between them.

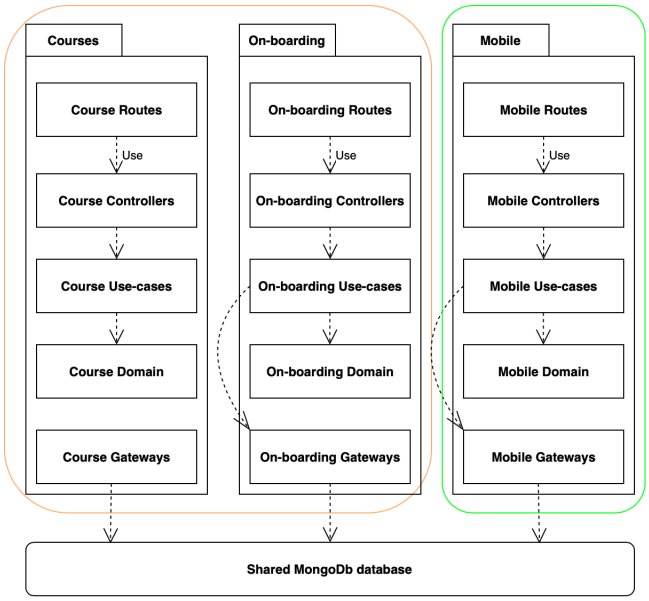


Figure 4.3: Component diagram of the back-end

End of collaboration

5 | Sprint #1

Duration of sprint: 22/09 - 06/10

5.1 User Stories

The first user story was to show a demo for the exercise screen to our PO

- *As a waste picker I want to view a demo course*

5.2 Sprint management

Based on experience, we have decided to split the group into subgroups with two members each and an extra member that would rotate between the subgroups and help where it is needed.

The group has also chosen to make a physical Kanban board as well as a digital Kanban board on ClickUp, to store the Sprint Backlog and its tasks, which was created by dividing the user stories into smaller sub-tasks.

Together with that, the group has also decided to utilise the website called Figma, which is a website for creating prototypes. This website would be used by the group to visualise their vision of given user stories that the PO would provide.

5.3 Cross-team coordination

During the first sprint, coordination was not USD between groups, because each group was in the start stage.

However, to agree on the general data structure of the problem domain, the group created a data model in collaboration with the web-oriented back-end group, because the two groups represented the two endpoints of the application.

After this, the back-end group Team-sharp-deluxe focused on the logins of content providers, while the mixed group Half-Full-Stack focused on the waste pickers login screen of the application and we, the front-end group Cows, focused on designing the exercise page.

5.4 Design & Implementation

In the first sprint, the group's main focus was to create a design for a "demo course". We figured that a course should have x amount of sections and these sections should then have multiple exercises. A figure of the imagined structure, which also aligned with the common data model can be seen below:

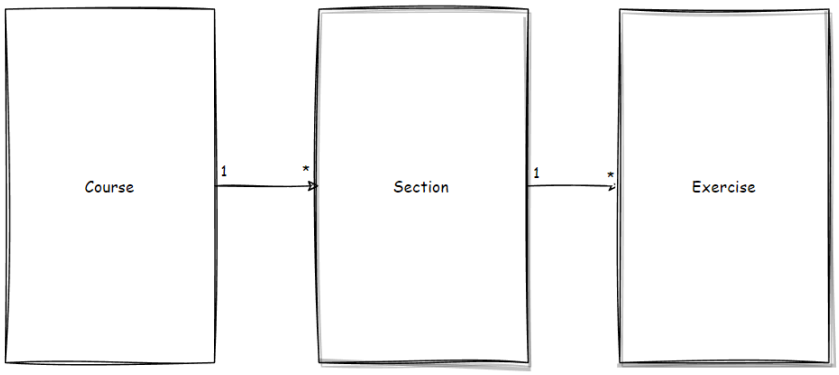


Figure 5.1: A sketch of the structure of a course

Our main requirement was to make an application, as simple as possible with a minimum amount of text and more focus on accessibility since the Brazilian waste pickers would not be well-educated and thus be limited by their ability to use the application due to a lack of reading skills.

Furthermore, the application should be engaging and exciting to use. Therefore, we introduced a gamified user interface, where the main inspiration came from the educational application Duolingo [14] and the online quiz application Kahoot (www.kahoot.com). Here, the idea of having a progress bar and a reward when answering correctly was implemented to improve the user experience and retention (see section 2.2).

The concept of how the structure of the content, created by the content creators, should be was based on the R2L theory mentioned earlier in the report (see section 2.3). It should be noted that the design can only encourage the content creators to produce content that adheres to the method, as it is ultimately the content creators that will decide the final form of the content. Nonetheless, the group designed the functionality of the application to enable the content creators to do just that.

The idea was that the amount of content for each exercise should be limited by using short videos to keep the exercises simple and engaging for the user. As further inspiration, Tik-Tok [15] where used, as the duration of the content is usually lower than 30 seconds.

Prototypes

Low- and high-fidelity prototyping was used by the group to generate a design that was shown to the PO to either be accepted or rejected. The process of making the low-fidelity prototypes started with each group member sketching their idea of an exercise screen, with the PO's requirements in mind. After this, each member presented their sketch to the rest of the group, after which each member had to vote on one of the sketches. This ended up like this:

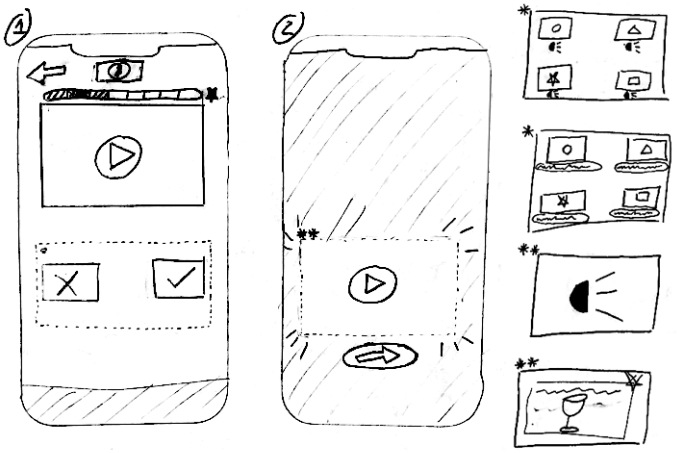


Figure 5.2: low-fidelity prototype of exercise page

After the low-fidelity prototype was accepted by the PO, the group went on to create a high-fidelity prototype. This process was done in a similar way to the low-fidelity prototype. Instead of each member making their own high-fidelity prototype, the group was separated into two groups that were going to make a prototype each. In our group, we used Figma (see www.figma.com), which is an online tool that is specialized in creating prototypes in a straightforward way. The result can be seen below:

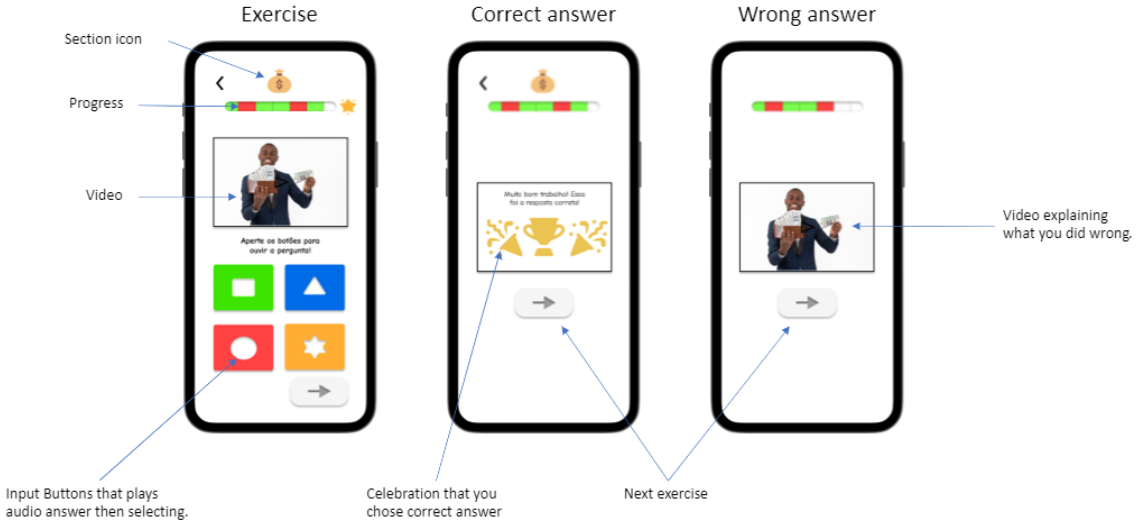


Figure 5.3: high-fidelity prototype of exercise page

Exercise screen

The group chose to start working on the "inner" layer of the course, which was the exercise screen. One requirement that the group had gotten from the PO, was that the exercises should be doable for people who were illiterate. Furthermore, the video would provide some kind of question, which the user could then answer by pressing one of four buttons. We took inspiration from the Kahoot quiz-like structure (see www.kahoot.com), especially considering its design. At the end of Sprint #1, we ended up with an increment where these features are implemented, as can be seen below:

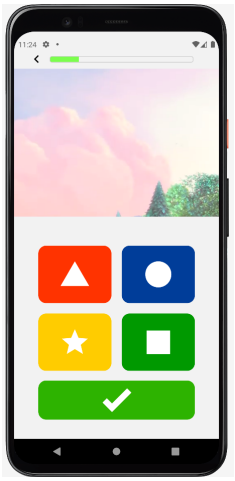


Figure 5.4: Exercise screen at sprint 1

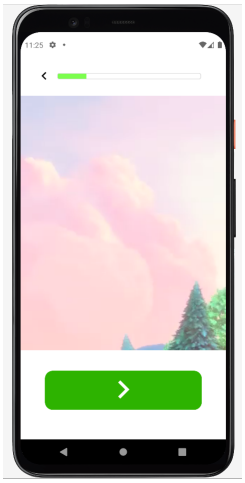


Figure 5.5: Wrong screen at sprint 1

5.5 Sprint Review

At the end of the first sprint, the user story was accepted by the PO. The first sprint's results were satisfying for the PO. The objective was to produce a representation of the content inside a course. Even though the group only managed to create an exercise screen, the PO was satisfied with the work.

5.6 Sprint Retrospective

The group discussed splitting up into smaller groups of two to three individuals to be more effective in working on smaller tasks. This method was effective in the implementation part of the sprint, but it was necessary for the group to be working together as a whole during the design phase to get everyone's opinion. Working in smaller groups went incredibly well when we had to code small tasks from the backlog because it was easy for two to three people to communicate and code together on a single screen.

Time estimation was used in Sprint #1 to attempt to figure out the size of the tasks on the kanban-board. Each member would think of the number of hours a task would take and all members would say it out loud simultaneously, we would then take the average number and assign it to that specific task, this helped with figuring out the workload of each individual sprint. If we had too many work hours, we would have to prioritise which tasks would be completed. This would of course happen in cooperation with the PO if it was the case.

For future reference, we made an agreement that we should have a supervisor meeting once a week.

Overall, the group had a very low absence rate, and everyone was quite engaged. There was enough time in the sprint for the group.

6 | Sprint #2

Duration of sprint: 06/10 - 20/10

6.1 User Stories

The demand for this sprint was to expand the application so that the user was able to have different sections with their own individual exercises. Therefore, the group made four user stories for this sprint.

The first user story focuses on the expansion of the application itself, as the user should not be bound to one specific exercise, but should be able to switch between different categories or sub-categories within the application.

- *As a waste picker, I want to have an overview of a course with all the sections inside*

User stories two and three lay the foundation for the gamification element of the application, as the PO wants the user to be more engaged in the content of the application by showing progress, as well as gaining achievements when completing exercises.

- *As a waste picker, I want to see my progress within a specific section.*
- *As a waste picker, I want to see my progress for sections in the course view*

The last user story is based on personalizing the application for each user. This is important to keep user retention high because the user will feel that the application is tailored to them and their needs.

- *As a waste picker, I want to see my profile(demo)*

6.2 Sprint management

The group had four different user stories, which compared to the first sprint, meant more splitting within the group to advance on all four user stories at the same time.

At the beginning of the sprint, we started out by deconstructing the four larger tasks into a lot of smaller tasks, so our physical Kanban board was filled with small and concrete tasks. When creating a sub-task, the group estimated how many hours, we believed the task to take, so that everyone had a united opinion on the level of difficulty of the task.

The Kanban board was used each day to get an overview of what each group was working on, what assignments that have not been started yet, and which tasks were completed already.

When a task was completed by either a sub-group or an individual, they already knew the next task inside their scope, as most of the assignments were gone over in the first meeting, when the Kanban board was created.

Afterwards, the group divided itself into smaller subgroups of 2-3 people, so each individual had at least one team member to talk issues over with. Due to members of the group being absent, and because of work or other obligations, the group made some "flex" tasks, which were mostly assignments related to the report, so that members of the group could work on assignments when they would have the time to do so.

6.3 Cross-team coordination

During this particular sprint, coordination across the three groups was not used because all groups thought, that they still were in the discovery phase. However, this turned into complications during the sprint, as no communication, but a lot of implementation, led to the project moving forward, but in three different directions.

Our group thought, that we had a clear indication of what the other two groups were doing, but when we were presented with their part of the product, it was not the same vision. Therefore, all groups had to backtrack a lot of progressions to make the implementations work together.

We did however talk about the overall vision throughout all three groups, which would be the personalization of a user. So the back-end group, who mainly focused on the database and the web application, focused on log-in for content creators, and the mixed group, who mainly focused on tying the database together with the mobile application, focused on the log-in screen for the waste picker in the application.

Our group, the front-end group, decided on the different aspects listed in the section before, but still with the collective goal of personalizing the application.

6.4 Design & Implementation

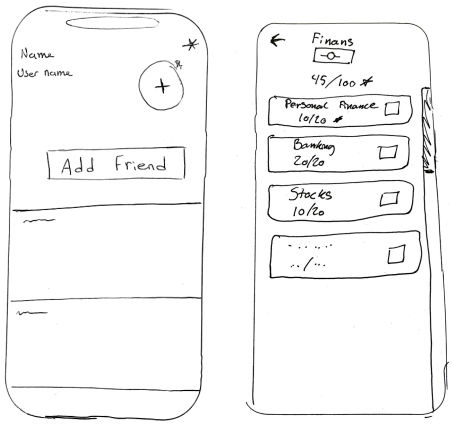
The focus in the second sprint were to make the course view and a profile view. These two segments provide information about the thought process of the application during the sprint, how the user interface evolved based on low-fidelity prototypes, and which widgets are used and why.

Prototypes

The group made low- and high-fidelity prototypes to create a design that got approved by the PO before starting the implementation.

We made two different low-fidelity prototypes: one for the profile screen and one for the course view screen (see figure 6.1). The importance of these screens is the same as the exercise view, which is to create a nice and easy overview while implementing gamification and personalized elements as well.

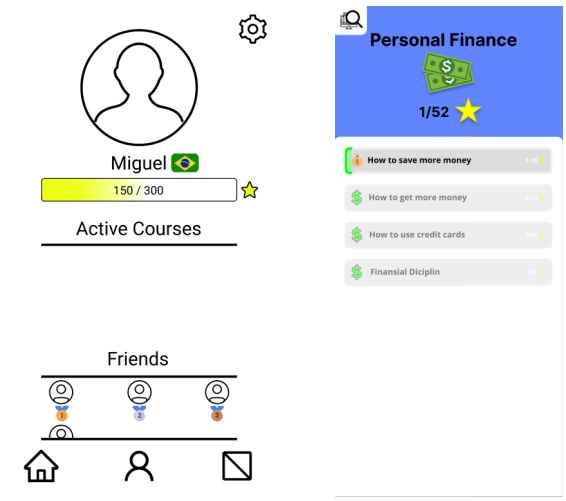
Each group member sketched a low-fidelity prototype for each of the screens. Below are figures showing two of the sketches, which represent a good baseline for the overall structure, that the group chose to move forward with.



Profile screen Course view

Figure 6.1: Low-fidelity prototypes for profile and course view

After the low-fidelity prototypes were constructed and a general design was accepted by the entire group, high-fidelity prototypes were made by each of the screens using FIGMA:



Profile screen Course view

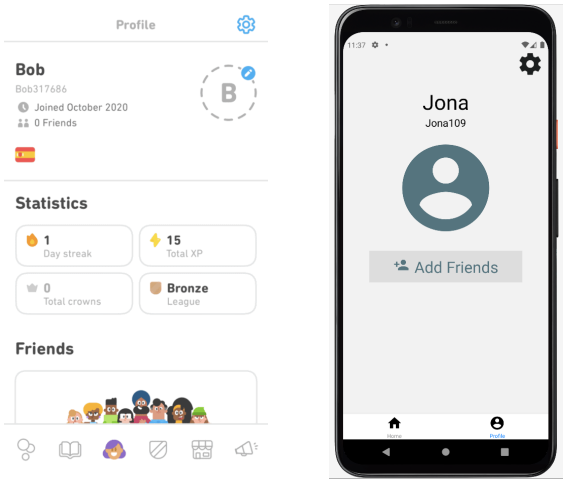
Figure 6.2: high-fidelity prototypes of the profile screen and the course view

Profile screen

When thinking of the design for the profile screen, the group's main focus was on getting a simple and easy overview, without too many features, that could confuse the waste pickers when using the application. When creating the demo for the profile screen, it was important that the personalized element was present from the moment the waste picker accessed the screen. This was made clear by the use of a profile picture in the centre of the screen, as well as the user's name above.

To create some form of gamification aspects in the profile screen, the group made a button to add friends, so the user would be able to see their friends' progression in the future when checking out their profile.

Figure 6.3 shows the current state of the profile. The initial application has a profile screen, so the group had to find other applications to draw inspiration from. As gamification and retention are key without using too much text due to the limitations of the project (see section 2.1), the group turned to Duolingo for inspiration. The problem with Duolingo is that the application has a lot of features that can seem unmanageable at first glance so the group decided to remove all the unnecessary details, to begin with, so the core features are present, and then maybe in the future implement more gamification elements. There is an image of the Duolingo profile screen for reference below as well:



Duolingo profile screen Current profile screen

Figure 6.3: The current state of the profile screen at sprint 2 compared to Duolingo

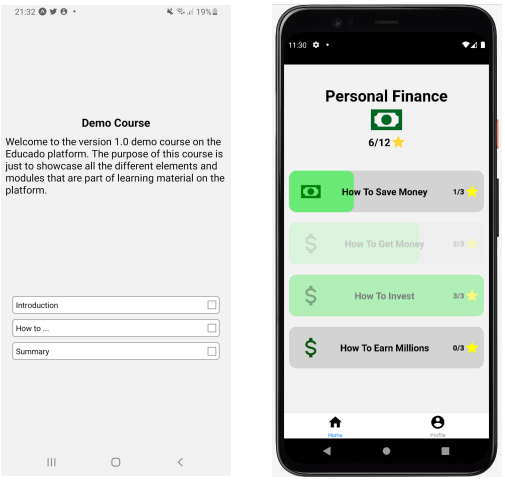
Course screen

The course screen showed a category name and an associated icon at the top of the screen to inform the waste picker which category they were currently in. Here, the user would also be able to see their progression across all of the courses shown on the screen. This progression was established as a fraction indicating progression denoted in a star currency. Like the profile screen, the main focus was getting a simple and easy overview, without too many features.

As mentioned earlier, the main focus for this particular screen was a simple overview. The user should be able to quickly realise what is what without having to click the different buttons on the screen actively. This was implemented by creating a list of the different sections, where each section has its unique title, and an icon to better understand the context of the section and the individual section progression. The latter was shown by using two forms of detailing, the first being the fraction mentioned earlier, but instead of it being the overall progression, it would only be for the specific section. The second form of progression detail is the colour of the section. As the waste picker progresses further, the section box would simultaneously becomes greener.

We also implemented some shading on the sections, as we wanted the user to be guided towards the sections that were previously progressed by the waste picker, although it is important to note, that all courses could be chosen.

Below is two images representing the initial course view received at the beginning of the semester and the newly implemented one. As mentioned earlier in the "Limitations" section (see section 2.1), the waste pickers are illiterate, so the limitation of text is necessary. The use of widgets is therefore important, hence the implementation of icons in the new course view, as well as the gamification elements, incorporated in stars, to keep engagement and viewer retention high.



Initial course view Current course View

Figure 6.4: Here is the new course view compared to the initial course view

6.5 Sprint Review

By the end of the second sprint, all four of the user stories got approved by the PO as the aim was to make a rough demo of the different screens, where the specific detailing of the screen could be made in a later sprint. The PO was pleased with the design of the interfaces, and thus the increment that the group had produced in the sprint.

The sprint planning for the next sprint was based upon the common goal between all three groups to integrate each other's work into a functioning working product.

The PO was keen on adding additional different gamification elements in the application so that user retention would be high and the waste pickers would keep engaging with the content.

6.6 Sprint Retrospective

When reflecting on the second sprint of the entire project, the group discussed the value of every member creating low-fidelity prototypes, as this way of doing things took up a lot of time at the beginning of the sprint and could perhaps have been done in two to three people instead of seven.

Another thing the group spoke about, was communication with the PO. Because of the many design choices in front-end development, it was important for us to have a direct line of communication where we would be able to get things continuously approved quickly, and not just at the end of the sprint. Therefore, we obtained the PO's WhatsApp, as this gave us the tool to communicate at all times.

Communication with the PO was not the only communication, that was needed in the upcoming sprints. This sprint was heavily affected by a lack of communication between the three groups, and this needed to change so that every team knew the direction and what ends needed to be connected with each other. The difficult part for all groups was the fact that nobody knew while being in the process, that communication was critical, therefore we sought out not to use it. In hindsight, there might have been some tools or exercises that would have been helpful to learn beforehand, as it would have made the progression throughout the entire project better.

At the end of the retrospective, the group talked about how the smaller subgroups had helped the productivity a lot, and how it would be an important tool to use in the upcoming sprints.

7 | Sprint #3

Duration of sprint: 20/10 - 01/11

7.1 User Stories

The next step in developing and expanding the application after getting an overview of the course with all the sections within, is the ability to explore new courses. Therefore, the goal of this sprint was for the user to be able to explore new courses and thus have a basic MVP (minimal viable product) with all important elements, such as sections, exercises, etc. So the only user story for that sprint was:

- *As a waste picker, I want to be able to explore new courses*

This user story also carries several tasks and decisions inside it, such as determining the number of stored courses in the course view.

7.2 Sprint management

In this sprint, we followed the same work flow as in the previous sprints. We started by dividing the user story into small and concrete tasks and put them onto our Kanban-board so we had an overview of the tasks we had in the sprint.

Then we made sure that each member of the group understood the tasks to be able to finish the wanted task. Afterwards, we started to assign tasks to sub-groups or one of the members, depending on how difficult it is. Then we made an estimated on how time consuming each task will be.

When a tasks was completed, it would be moved to the done part and another task would be started, where each update would be represented on the Kanban-board.

7.3 Cross-team coordination

In this sprint, the cross-team coordination did not happen as it was supposed to. The three groups kept in contact but not as originally agreed upon, which led to confusion in the groups. The dependencies between the groups was growing due to the progression of the development. Therefore, coordination between groups was much needed to make sure all groups were on the same page.

We did talk about the common goal of this sprint, which was a working product with a working login system. So the mixed group focused on the login system and the password strength and the back-end group focused on adding some more features to the content creator applications and dashboard. Our group focused on the course view and making the features within the application dynamic.

7.4 Design & Implementation

This section provides information about the thought process of the application during the sprint, how the user interface of the explore screen evolved based on low-fidelity prototypes, and which widgets are used and why.

Prototypes

The group has made low-fidelity prototypes that would show the explore screen to the PO to get the design approved and then start working on the implementation part.

On the contrary, in the search screen in the initial state (see figure 1.5), we have discussed reducing the number of texts so the application becomes more user-friendly. Therefore, the title of the screen will be a single word in Portuguese "Explorar". And to make the application easier to use we decided to add filtering to the courses, so the user will be able to press and filter the courses by categories. But this feature would not be implemented in this sprint, since we just focused on exploring new courses. The next step in making the explore screen is to decide where the active courses should be. Active courses mean the courses that are in progress. So we decided to place the active courses at the top of the course list and give them a colour, so they can be distinguishable.

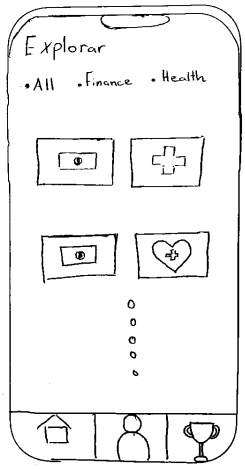


Figure 7.1: Low-fidelity prototype for explore screen

Explore screen

After the low-fidelity prototype was approved by the PO, we started making the implementation of the explore screen. The implementation was based on the low-fidelity we made, with some additional changes. We started by adding a button in the bottom menu bar with a search icon to make it easier for the user to get on the explore screen. The title changed to "Explorar Novos Cursos" which means "Explore New Courses" to improve the user interface. The three filtering buttons were placed under the title, where we have the current topics in the app. Then the list of courses was implemented, where the active courses are placed at the top of the list and the rest of the courses come after. We decided to give the active courses the colour green so the user can see the difference between them and the other courses. Also, we decided to give a small title for each course plus an icon to show what the course is about. As mentioned in section 7.4, we did not finish implementing the filtering system, therefore we continued with the implementation of this in the next sprint.

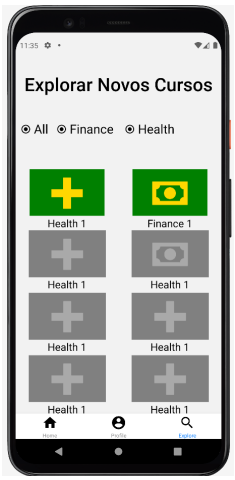


Figure 7.2: The state of the explore screen at sprint 3

7.5 Sprint Review

At the end of this sprint, the user story was accepted by the PO, which was primarily about the explore screen, while the part about storing the courses ended up not being our responsibility. We faced some issues while we tried to make the different features dynamic in the application and especially when we tried to make the video in the exercise screen dynamic.

The sprint planning for the next sprint was based on refactoring and moving closer to the final working application.

7.6 Sprint Retrospective

This sprint we were missing some members due to sickness. In addition to that, we spent time searching and trying to solve the issues we came across while trying to make the functionalities dynamic, thus there were not a lot of increments.

8 | Sprint #4

Duration of sprint: 01/11 - 15/11

8.1 User Stories

The main focus in this sprint was to get a course shown using the stored information on the phone. This would allow us to make an application that reflected a course created by content creators. During this, we have seen the group who worked with the web application for content creators use a UI library and we wanted to implement something similar. Therefore, a secondary focus was to shift attention to the user experience. The following two user stories were made for that:

The first user story focuses on visualizing the stored course as a waste picker.

- *As a waste picker, I want to be able to view a stored course*

The second user story would set focus on a uniform design by using a UI library.

- *As a waste picker, I want to be able to have a better user experience while using the App*

8.2 Sprint management

The group continued to work with the two Kanban-boards where the user stories are broken down into smaller tasks, after which they will be assigned to a subgroup.

8.3 Cross-team coordination

In this sprint, we made a breakthrough in the importance of cross-team meetings since we started to heavily depend on the other two teams in order for our front-end to reflect the data in the back-end and the data that needed to be stored on the device. We started to communicate the idea visualized in Figure 8.3 by the middle of this sprint.

8.4 Design & Implementation

The main idea of the fourth sprint was to get the Exercise screen from Sprint #1 (see figure 5.4) to also work together with Course Screen from Sprint #2 (see figure 6.4) and Explore screen from Sprint #3 (see figure 7.4). The group chose to focus on getting the Course screen to show a course that was actually stored on the device of the user, while also enhancing the user experience with better UI. We did not make any prototypes for this sprint, as the user stories focusing on design for this sprint could be done faster by implementing design directly with react. By playing around with different libraries and seeing how they would look in the application we could move faster. Design choices for the screens will be explained in their corresponding segments.

Course screen

The Course screen was the first screen to get a rework based on the idea of having an application with a better overall UI experience. The Course screen had to be changed a lot if it was to follow the same feeling as the rest of the app. In this current stage, we decided to only have the UI reflect the data that could be stored in the app. This meant that all hard-coded progress indication was removed. The result can be seen in figure 8.1 and when compared to figure 6.4 from Sprint #2, the only thing we kept were the Course Icon, Title, Section Titles, and Section icons.

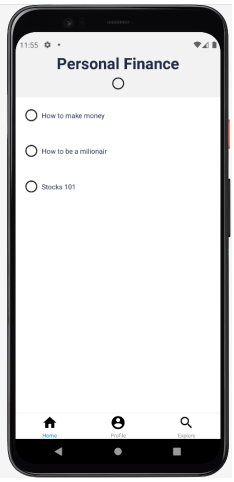


Figure 8.1: The state of the Course without Active course at sprint 4

Explore screen

The Explore screen started to take shape with both better UI as seen in the lower part of figure 8.2 while we also tried to make the initial infrastructure in order to support the application to base its content on stored courses from the actual device.

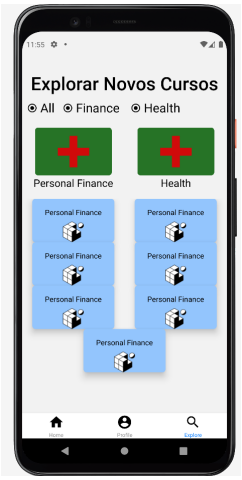


Figure 8.2: The state of the Explore screen at end of sprint 4

The initial communication that needed to be made from front-end to back-end started to take shape as seen in figure 8.3. The idea was that an element on the screen should directly reflect the state of the application based on what was stored. The idea of having a controller or service layer in the application that took care of the state of storage came to life. And this was where the true separation between the front-end group and back-end group also became visible. We decided that the controller/service layer was a back-end task although the code would still be part of the application and shipped inside the app.

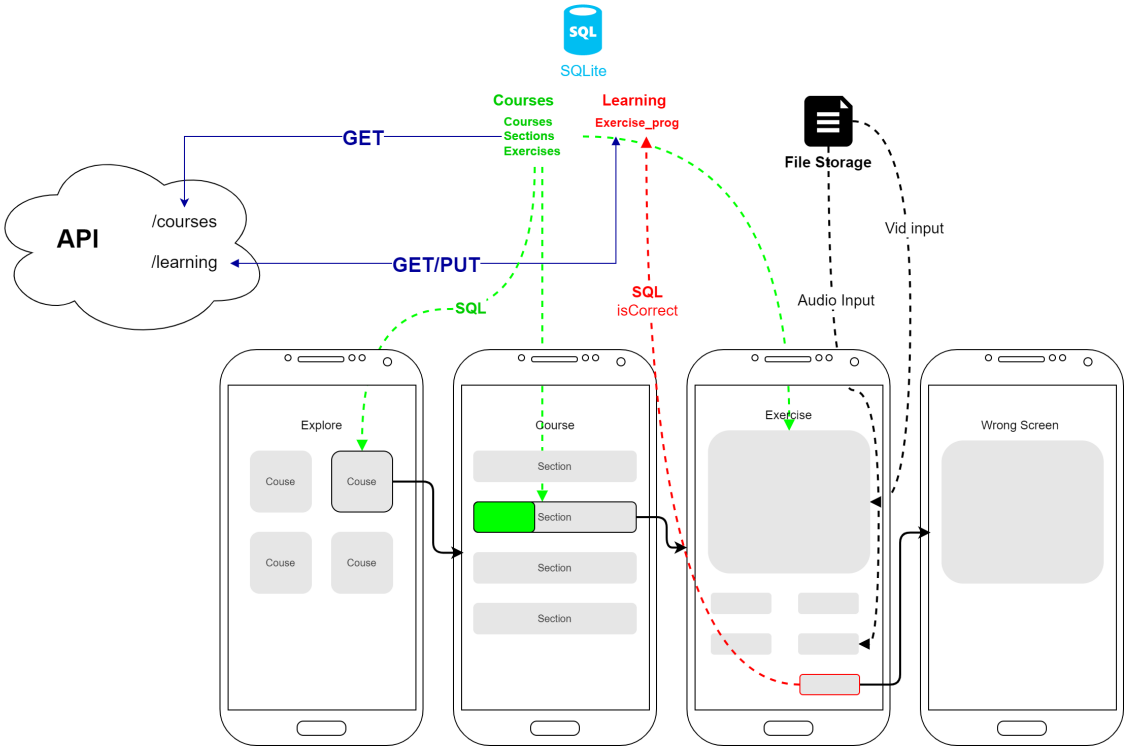
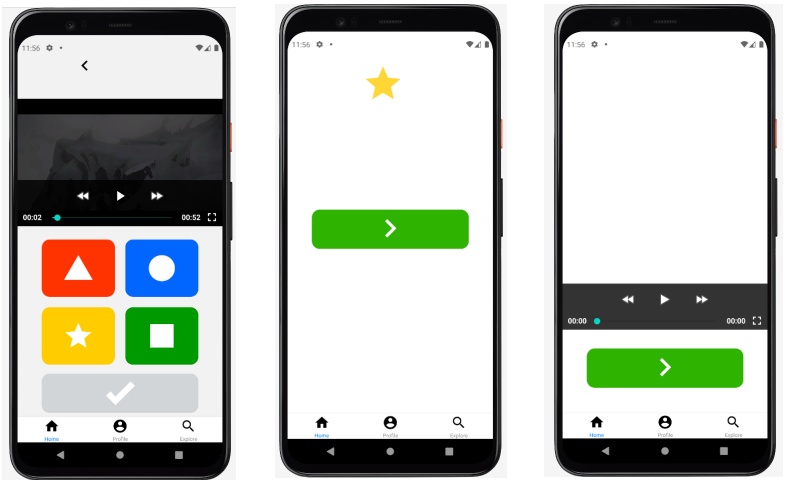


Figure 8.3: Technical picture of communication between front-end and back-end

Exercise screen

In Sprint #4, the Exercise screen was made to be able to navigate to two different screens based on the choice of the button clicked (see figure 8.4). If a button marked with a correct answer was clicked, the 'Right Answer Screen' would be navigated to. If a button marked with an incorrect answer was clicked, the 'Wrong Answer Screen' would be navigated to. An option on both of these screens to click the green button in order to continue would be present, and this button would navigate to the next exercise in the current session if it was clicked.



Exercise Screen Right Answer Screen Wrong Answer Screen

Figure 8.4: The state Exercise Screens in sprint 4

8.5 Sprint Review

We found out that the user story about being able to view a stored course was not entirely our responsibility, meaning that we did not sufficiently delimit the responsibility of each group when the user story was defined. This resulted in us making some breakthrough thoughts that we could bring to the next meeting between groups, but unfortunately, we did not have anything to contribute to the sprint goal with this user story since we used a lot of time thinking about how we could communicate from front end to this storage layer that did not exist yet. This user story ended up being way more significant than we originally thought, and hence we knew beforehand that it was going to get rejected, and we extended it into the upcoming sprint. The user story about having a better user experience also ended up taking longer than we anticipated, which meant that we knew this one was also going to get rejected in the sprint review.

In this sprint, we made a breakthrough about how the local storage would be implemented, and what the responsibilities between each team would look like. For this reason, we started to feel the need to communicate closely with the other groups. Therefore, we talked with the other groups about having alignment meetings between the scrum masters of the different groups to get a better understanding of where each team was at without involving everyone. This way only one person needed to make sure that information was flowing between teams.

8.6 Sprint Retrospective

Group retrospective

In the retrospective, we talked about how we could have more time to work at our jobs while being fair to everyone in the group about how many hours were put into the project. We had previously worked with flex time, and this idea was initially a good one, but the people who had a lot of flex time started to go into debt and did not have a lot of time for the project, and we could see that it was not sustainable to keep racking up more flex time as we went along, so we tried to eliminate it from the schedule and essentially just gave everyone who was not working a job half a day off so that the work balance between group members where aligned.

During the retrospective, we also made some more clear guidelines as to whether we meet whenever course/lectures are moved in a week: We ended up agreeing to use the time on the project if a course was moved, as the time reserved by the moved course would affect the project's timescale in the future.

We also talked about how to communicate more when we are absent from the group project. If you are sick, you need to tell the group the first day and tell them the day before you come back. This way everyone always knows who will come work on the project on the given day and rather wait for someone before starting on something big. Furthermore, the group felt like we spent a lot of time on research and thus not a lot of increments in code.

Shared retrospective

During this fourth sprint retrospective, we did a collaborative exercise with the other groups.

Written in collaboration with all groups

During the sprint retrospective, at least one member from each group formed a team and discussed what they felt did not go well in this sprint and how we could improve it. A summarized list of some of these discussions has been collected below:

- **Sprint review:** One of the things discussed was the flow of the sprint review. It has happened that during one group's review they have been interrupted with questions or comments. The groups should be allowed to finish their review before questions are asked, and ideally, further discussions about future approaches should be done after the review.

  There has also been a noticeable lack of engagement from other teams during the presentations. Everybody should follow along during the review.

- **Communication:** There is still insufficient communication between the groups. There has been a slight increase in meetings since it was discussed in earlier sprints, but it has mostly either happened spontaneously, or only two of the three groups have been partaking in it.

  As per request, the number of meetings between groups has increased slightly. However, the meetings have been quite informal as they are usually not scheduled ahead of time and occur in a more spontaneous manner. There has only been held one official meeting in which all three groups partook.
  In order to get a better communication flow, a fixed meeting between all Scrum Masters of each group has been set. For the rest of the project, the groups will have at least one meeting every Friday at 10:00.

- **Backlog:** A topic that has been discussed several times during this project is having a shared backlog between all groups so that every group always know what the other groups are working on. The PO initially promised to create this with the Trello boards that were shared with him, but we have not yet received them. A person from each group should get together to set up a backlog. Ideally, the Scrum Masters will do it at their weekly meeting.

- **Alignment:** Each group has been focusing on their respective user stories and goals. As a result of this, the common end goal has not received a lot of attention. This also means that everyone has made their own design choices for both the code structure, as well as the front-end design.
  This has led to extra work, as the design patterns should ideally be consistent in a repository so future developers easily can understand the flow of the code.

  There also needs to be better alignment between front-end and back-end developers. If changes need to be made to one of the routes in the back-end repository, a front-end developer has to be informed about this, since it can affect their code as well. Another contingency that will be used from now on is to have shared documentation for all REST APIs, so front-end developers can easily check existing APIs and how they are to be utilized.

- **Definition of Done:** One way to fix our alignment issues would be using a common definition of done (DoD). There has not been any agreed-upon set of rules for this yet. One DoD that will be added is that each repository should have a branch that will mock the master branch of the project. Only the working code should be pushed to this branch. In order to maintain this, each group should appoint a code review master. Their job will be to code review and accept the other group's code before it will be pushed to the master branch.
  When the code is reviewed it should also be checked to see if it follows the newly agreed upon design pattern, so our code base is aligned.

End of collaboration

9 | Sprint #5

Duration of sprint: 15/11 - 29/11

9.1 User Stories

In the previous sprints, the group have been focused on the functionalities of the app. Eventually, this led to a visually impaired application. For this reason, the PO decided, in cooperation with the group, that it was time to focus on making the application more visually cohesive. Which led to the user story:

- *As a waste picker, I want to use a visually cohesive app*

Another crucial aspect of the application is its user-friendliness. In addition to being functional, the application should be intuitive and easy to use. To ensure that it meets these criteria, the application will be tested by Brazilian waste pickers (see section 11.2). The final user story ended up as follows:

- *As a waste picker, I want a user-friendly experience*

In the previous sprints, we were able to show courses with a StorageController, we did this to mock some data so that we could design the application. In Sprint #5, another focus was to go from this mocked data to an actual local stored data, which led to the user story:

- *As a waste picker, I want to view a downloaded exercise of a section*

9.2 Sprint management

The group will continue to work as described before in the previous sprints. We ideally want to create time to write tasks revolving around the report that has to be written at some point, and we are aware that we are behind schedule.

9.3 Cross-team coordination

In the previous sprint's retrospective (see section 8.6), it was determined that effective communication between teams was crucial for the success of the project. As a result, a plan was implemented where a Scrum Master from each team would meet every Friday to document the progress of each team. This helped to foster the development of a shared, cross-functional product. In the fifth sprint, our team was required to collaborate with the back-end mobile group in order to successfully implement the user story outlined in (9.1).

9.4 Design & Implementation

One of the focuses of this sprint was to enhance the design (see section 9.1). The majority of the group's work was to make the application more user-friendly and make the design cohesive. Furthermore, we had to make it possible to locally store exercises. This was a "shared" user story since a big part of the responsibility for its success was on the back-end mobile group. Our group had already prepared for this by mocking the data. We only needed to adjust a few things to make it work.

In the fifth sprint, we decided to add tailwind CSS see (https://tailwindcss.com/) to our project. Tailwind CSS is a framework that makes it easier for developers to write valuable CSS code. Tailwind CSS makes it possible to create your own theme throughout an application. This helped us produce a matching design in the application.

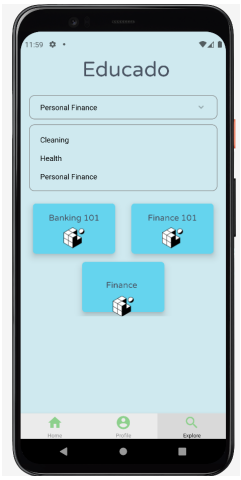
```
theme: {
  extend: {
    colors: {
      cyanBlue: '#65D4EE',
      limeGreen: '#9DE89C',
      yellow: '#FAC12F',
      babyBlue: '#CFE9EF',
      limeGreenDarker: '#8DD08C',
    }
  }
},
```

Figure 9.1: The project theme colours in tailwind CSS

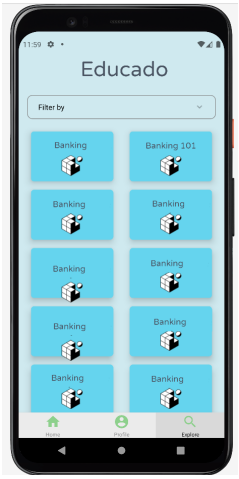
Another helpful design tool we used was https://www.canva.com/colors/color-palettes, with this tool the group choose a palette of colours that matched together. We ended with a palette called "**Summer Cocktail**" because we thought they represented the colours of the Brazilian flag.

Explore screen

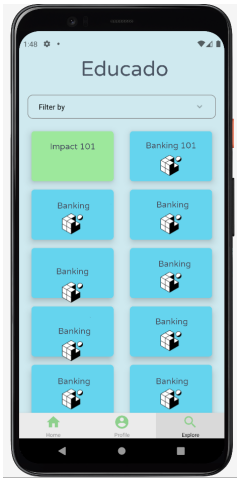
The explore screen was re-designed in Sprint #5. Instead of separating the active/non-active courses from each other, we decided to colour code and filter them instead. This gave the design a more unified look. Furthermore, a filter function was added to the screen, so a user could filter the courses by category. The list is made in a way so that the active courses always will be at the top of the list. This makes it easier for the user to find the active courses.



Explore screen (filter function)



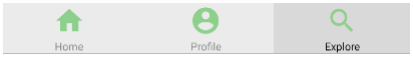
Explore screen (no locally stored courses)



Explore screen (one stored course)

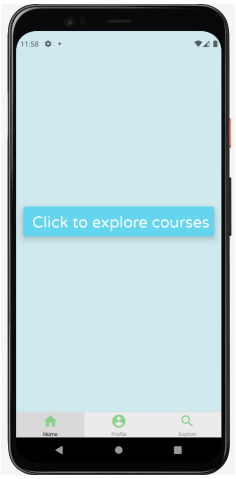
Navigation Menu

The navigation menu was styled with shadows and theme colours.



Course screen - with no active course

The course screen was changed so that when the user loads in for the first time, the user will be prompted with a button to go explore courses.



No active courses screen

9.5 Sprint Review

In the sprint review in Sprint #5, only one of three user stories was accepted. A reason for this could be, that a lot of the group's time was used to merge and align with the back-end mobile group. Furthermore, the user stories (see section 9.1) relied on the progress of the back-end mobile team.

- Accepted user stories
  - As a waste picker, I want to use a visually cohesive app.
- Rejected user stories
  - As a waste picker, I want a user-friendly experience.
  - As a waste picker, I want to view a downloaded exercise of a section.

9.6 Sprint Retrospective

The main focus of the fifth retrospective was communication and alignment with the other groups. The majority of things that went wrong in Sprint #5 could have been better/fixed with good communication and better planning between the groups.

User stories

1. User stories were not specific enough/too many

The initial user story workload of Sprint #5 was very overwhelming, and the group did not think we would be able to finish them all. The user stories were also phrased in a way where the acceptance criteria can be very subjectively measured (see section 9.1). It is hard to define, what a visually cohesive and user-friendly application is. This led to a stressful experience for some group members. In retrospect, we should divide these user stories into smaller more feasible user stories. This would have given the group a feeling of progress and a better understanding of the workload.

Alignment & Communication

Alignment and Communication in the fifth sprint were essential for the success of the user stories since this user story involved two groups.

1. Merge happened late in the sprint because of a lack of alignment and communication.
    - Due to bad/no planning the merge between the front-end mobile group and back-end mobile was delayed. If we had used more time planning at the start of the sprint, we would have saved time and work.
2. Multiple merges across sprint.
    - We used a significant amount of time merging with the back-end mobile group, this time could maybe have been saved had both the groups been aligned.
3. Knowledge-sharing in groups and between groups after merging to ensure that everyone knows the codebase.
    - Due to the feeling of being behind schedule, this process was down-prioritized. In a perfect scenario, every member of each group should know the code.
4. Lack of common Definition of Done led to coding style issues and UI-styling.
    - During the sprint sessions with the PO the groups would agree on user stories for the sprint. An improvement could be to agree on the definition of done in these sessions. This would also solve the problems with the unknown workload of the user stories (see section 9.6)
5. Frustrating to not have things work due to the late development of features from the back-end mobile group.
    - The back-end mobile group were lacking behind, and this ended up impacting us since we relied on them to work on their part of the user story. Our group tried to poke them to get them going. This was very frustrating since there was nothing our group could do, this meant that the user story went to a halt.

Backlog

1. Sprint backlog is used less than before, lack of tasks one can start.
    - This problem could be solved if the user stories were defined better and spitted into smaller more doable tasks (see section 9.6).
2. Tasks that did end up in the sprint backlog were nicely specified. Everyone knew what the task was about.
3. ClickUp https://clickup.com/ was not being used.
    - ClickUp is an online kanban/planning tool for agile development. This tool could have been useful in Sprint #5 since there was a lot of absence in the group. It could help give an overview to the members who were absent.

Positive experiences

1. It was a success to back-sit code with the other groups.
    - A good learning experience was sitting and coding with the other groups, this also helped us in understanding the other group's code. We learned about their struggles and ideas. This also helped us align.
2. Every time we had a merge with another group one member from each group would be there to review the process.
    - This helped make sure no mistakes were made. Furthermore, it gave the opportunity to show and tell about each individuals groups code.

Other issues

- Tailwind (if we implemented it easier, it would have been easier to switch the styling library) has a lot of "quirks" on Android.
- Stressful sprint, we forgot some of the structural elements in our workflow
- Because of dependencies, the group did not manage to create time for report-related tasks
- A lot of absence due to being sick/work.

10 | Sprint #6

Duration of sprint: 29/11 - 13/12

10.1 User Stories

The last sprint of this project was mostly used to fulfil the MVP (minimum viable product) and have a product ready for deployment. The first user story for this sprint was:

- *As a waste picker, I want to view a downloaded exercise of a section (extended from the previous sprint)*

This is a continuation of the same user story we had in the previous sprint, which we did not fulfil. The idea behind this user story is still the same; to integrate the application and give us the ability to store the courses locally by downloading them. The next user story was:

- *As a waste picker, I want to click the buttons in an exercise to hear the different answers (focus on text-to-speech)*

This user story would allow the user to hear the answers when they would press the answer buttons in an exercise. The last user story:

- *As a waste picker, I want to see my progression within a section (smaller focus on progression implementation)*

This user story is meant to visualize the progression within a section for the user, so they would know how far they are within it and how many exercises are left until the section is completed.

10.2 Sprint management

In this sprint, we kept working the same way we used to in the previous sprints. We began this sprint by dividing our user stories into small tasks and putting them onto our physical Kanban board.

These tasks will be done within smaller subgroups within our group, just like in previous sprints, unless the task revolves around integration. If it does, then we will make a subgroup that consist of a mix of our front-end application group with the back-end application group. This will greatly ease the process of integration, which will allow us intertwine two different parts of the application together, and we need both groups to cooperate to have it working correctly. Once the task is accomplished, we would naturally move it in our physical Kanban board to the done section and the subgroup will move on to the next task.

With each Daily Scrum, we will briefly go through the progress that each subgroup accomplished the day before and what the plan is for today.

10.3 Cross-team coordination

In this sprint, we continued the cross-team meetings where Scrum Masters would meet on each Friday and talk about the status of each part of the project.

During the meetings, we would ask each Scrum Master about what the team is working on, what is needed from other teams, what we plan on doing for the rest of the sprint.

Because this is the last sprint in the project, all teams were heavily focused on finishing what was missing to achieve the MVP and begin working on the rapport. From previous sprints, we chose to have a common layout for the rapport that each team would use. There were not a lot of code-related discussions going on as this is the last sprint, so the only thing we needed to do is integrate each separate part of the code into one cohesive application. A lot of time during the meetings was used on talking about how the common layout for the report should look like.

10.4 Design & Implementation

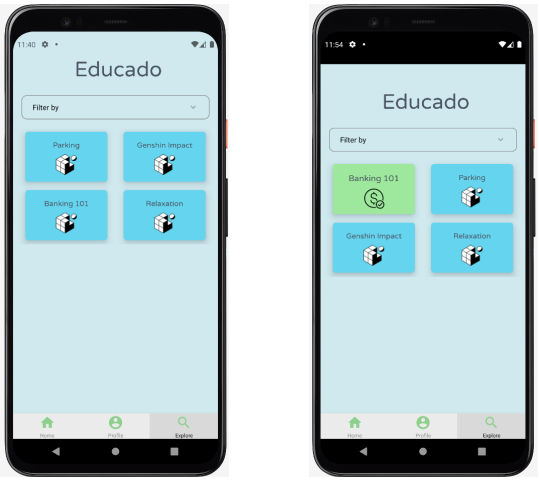
In this sprint, there was not much increment in relation to the front-end of the application. Therefore most of our implementation came from polishing of what was already done, what was still missing (was not dynamic), and the overall integration of the product. That is why there is will be no section about prototypes because we were not developing anything new in this sprint.

One of the things that were implemented in this sprint was making our text-to-speech system dynamic, which would utilize the expo-speech API [16]. Before we had our answers hard-coded into the app, but this time we can use the answers given by the content creators. This information is stored locally on the phone when the course is downloaded. The way our text-to-speech work is relatively simple, we just pass the specific text we want to play when the button is pressed. The code can be seen below. It is also worth mentioning that text-to-speech is also functioning offline, which is important for our users.

```
/  function handlePlaySound(tts) {
    Speech.stop()
    Speech.speak(tts, {
      language: "portuguese",
      voice: "com.apple.ttsbundle.Luciana-compact"
    }
    )
}
```

Explore screen

Besides this, we did a lot of integration of the app. Where the data would be used to dynamically implement the front-end. In figure 10.1, we can see what the application looks like with four different courses, and what happens when one of the courses is downloaded to the local storage, and how it changes visually.



Explore Screen with no stored courses

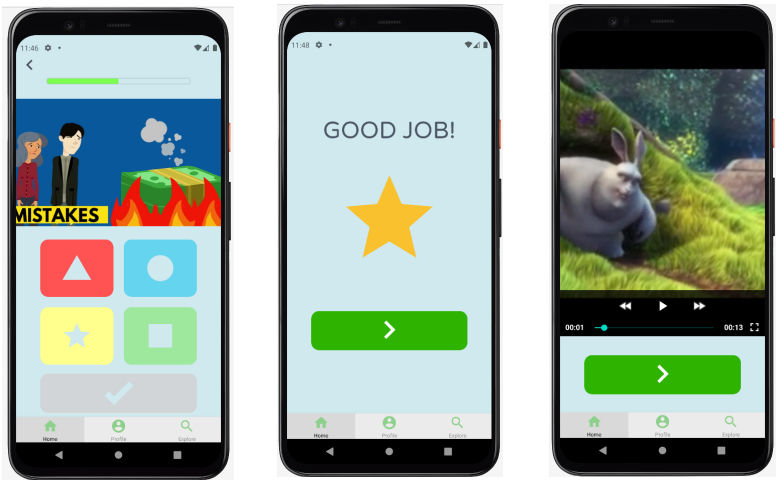
Explore Screen with one stored course

Figure 10.1: Explore screen without and with a stored course

Exercise screen

Besides this, we can also look at how the Exercise screen has changed since the prototype from Sprint #4, which was presented in section 8.4.

The present version of the application has changed a bit since Sprint #4. The present version can be seen in figure 10.2.



Exercise Screen Right Answer Screen Wrong Answer Screen

Figure 10.2: Present Exercise Screen with Right Answer Screen and Wrong Answer Screen

As we can see from the figure, the screens have changed a bit from the previous iteration. In the present one, the screen got a background colour and the colours of the buttons have also slightly changed. Which differs from the previous iteration. Besides the visual aspect, the Exercise Screen can now also dynamically change videos using the ones that are locally stored when the course is downloaded.

Course screen

The Course Screen has had a lot of drastic changes since its first iteration in Sprint #2 (see figure 6.4). The individual sections do not contain any icons or do not have the section progress that we had in the previous iteration. Besides those changes, we also removed the gamification elements, as there was not enough time to accomplish it and achieve a satisfactory result.

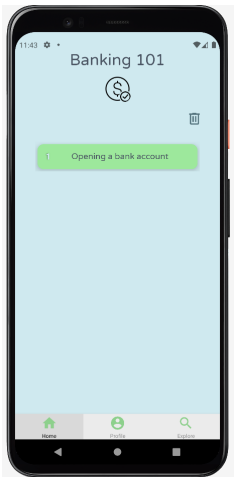


Figure 10.3: Course Screen

Profile Screen

When we look at the old Profile Screen (See figure 6.3) and compare it to the current iteration the changes were not as drastic as they were in the Section Screen. The main change was removing the friend list which was not doable at the present stage of the project. Besides that the option to log out and delete the account was added and settings were removed. The present profile screen is really simple the user can see their username, and phone number and have the two options that were mentioned before.

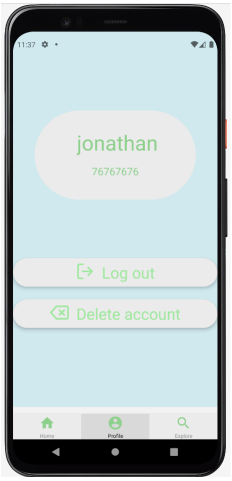


Figure 10.4: Profile Screen

10.5 Sprint Review

This sprint review was a bit different than the previous ones, it differs because this is the last sprint. Therefore, during the review, the increments of the sprint were shown with a common presentation where the overall flow of the application was presented. From the three user stories that we began the sprint with only two were accepted. The first one relates to the overall integration of the app, and the second relates to the text-to-speech aspect of the app, which also required the integration to work correctly. The last story which required a visual representation of the user's progress within a section was not fulfilled. Which meant that the user is not able to see how many exercises are done. Although the data is not visually presented in the application it is still being saved.

10.6 Sprint Retrospective

Group retrospective

In the Sprint Retrospective, we discussed how time-pressured we were during the whole sprint. Because there was a lot of stuff missing and how we were unable to follow our initial plan. We also had some framework-related issues where we could not test our front-end properly. Another thing is not being able to test the usability of the application on our target group, which is important to know if the application can be used by waste pickers. Although this is something that we plan on doing on the users in Brazil during the trip. The experience from previous sprints allowed us to be better at working with the other groups, we were more proactive with each other and we were able to keep up with our work schedule. This meant that we were able to fulfil the MVP that we set for ourselves.

Shared retrospective

Like we did in Sprint #4 (see section 8.6), in the sixth sprint retrospectives we also did a collaborative exercise with the other groups.

Written in collaboration with all groups

We began the sprint retrospective with an internal group discussion about Sprint #6. Afterwards, each group presented their thoughts on this sprint. This included discussions about what did not go well, but also about improvements from previous sprints.

- **Sprint review:** In the sprint review, it felt more like one product compared to the previous reviews, as when we began showcasing the web application for the content creators, and afterwards, we presented the shared work of the two mobile application groups together. Here we could also see the new course created during the web presentation.
- **Workflow:** It has been a very stressful sprint. Not only did we want to try and get a working MVP before usability testing in Brazil, but we also had to try and fix the issues that were presented from the Static Code Analysis. We did have technical debt from previous sprints, which ideally should have been continuously fixed, but at the beginning of the project, the focus was on implementing new features and learning to work together in order to ensure a good product.
- **Communication:** The communication in this sprint were a lot better compared to previous sprints, but there is still room for improvement.

Final collaborative retrospective

After the retrospective for sprint 6 we completed the development process with a final retrospective that covered the whole project from beginning to end.

- **Progress:** One thing we discussed was how we as students had evolved since a similar project in the 3'rd semester. Here we had to work as a single group that solved a real-life problem for a company.
  In 3'rd semester we had to spend a lot of time trying to figure out how to use the skills we learned in courses such as system development. Whereas in this semester these skills were simpler to apply, which was a great indication of our own development.
- **Sprint zero:** It was discussed that one way to start with a stronger foundation for common goals and better communication would be with a sprint 0. This was not something we were aware of and something that could be done before the end of sprint one. In sprint zero we should have not touched any of the code, the focus should have been on communicating with the other groups and agree on a common set of requirements based on the project we received and the wishes of the stakeholders and PO.

- **Communication:** One of the common topics throughout the development process was the lack of communication. This includes communication among the three groups, but also communication internally.
  Each group's sprint backlogs were not shared with the other two groups. This caused issues as we did not always know what the other groups were working on. Especially in the beginning when we had yet to establish a good communication flow.

  The lack of communication led to issues where there were risks of groups working on similar tasks, but there were also examples of different coding approaches in the same repository. In order to make it easier for future developers the naming conventions and approaches should have been discussed before we began the development.
  But, in the last two sprints, the groups started to work more as one unit with one common goal. People from different groups would start sitting together to fix common issues. Also, the communication between the front-end and back-end became clearer. If there was something the front-end or back-end needed from each other they would talk together about it to try and find a good solution.
- **Cross-collaboration:** In continuation of 'Communication'. We did not have any good tools that could help ease the process of working together across teams. This is of course one of the learning goals for this project, and we did in the last three sprints have a weekly meeting with the Scrum Masters from each team. But, one thing that was talked about was that we had hoped in the early part of the development process to be introduced to a set of tools that might have been useful for working with multiple teams on one project.
- **PO:** When the project first began, it felt as if the PO was learning his role along with us. Especially in the first few sprints, we were still not clear about a lot of fundamental knowledge from agile, so we had hoped that the PO could be of assistance. At the end of the third sprint, we had a guest lecturer joining us for the sprint review. The guest lecture came with a lot of useful feedback, that helped us, but also was of assistance to the PO. After this, the PO seemed to be more confident and aware of his role. This along with our expanded agile knowledge made sprint reviews and sprint planning easier to complete.

  Another thing that was discussed in the retrospective was that we would have had great benefits from the PO writing the user stories or setting clear sprint goals for each sprint. We wrote our own user stories and for the first few sprints, there was no agreed-upon goal for what should be achieved. This is one of the reasons why we started to accumulate technical debt.
  It was sometimes not easy to get proper approval for the user stories we had written for each sprint and at times it felt as if the PO was not aware of what was in the sprint backlog.
- **Sprint review:** Just as with the cross-communication, it was discussed that it would have been beneficial to have learned how an actual sprint review is normally handled. There was confusion about how to actually proceed with the sprint review, and it was not until the final sprint that it felt as if the review was about one product and not three different products.
- **Pipelines:** It was not until the last two sprints that we learned about pipelines and Continuous Integration. It is normally considered good practice to create a pull request instead of merging code directly into the main branch of the project. It was discussed that it would have been very beneficial for us if we were provided with an option and guideline to set up a pipeline that forced us to do pull requests.

Overall there have been a lot of frustrations during the project for every group. There was a very steep learning curve as we not only had to take over an existing project but also had to work together across teams.
In the end, our way of working together still has room for improvement and the product may still have some issues, but through the frustrations and failures, we experienced throughout the project we also learned valuable skills that we could slowly start applying while working on the project.

End of collaboration

Part III

Testing & Evaluation

11 | Testing

11.1 Software Quality Management

In this section, we will talk about how we tested the quality of our software using a code analysis tool called CodeScene.

CodeScene

To ensure and check the quality of our code we have used CodeScene to analyze our code and look for potential risks.

The results were outstanding, in the figure below, we can see a map of the different parts of the code.

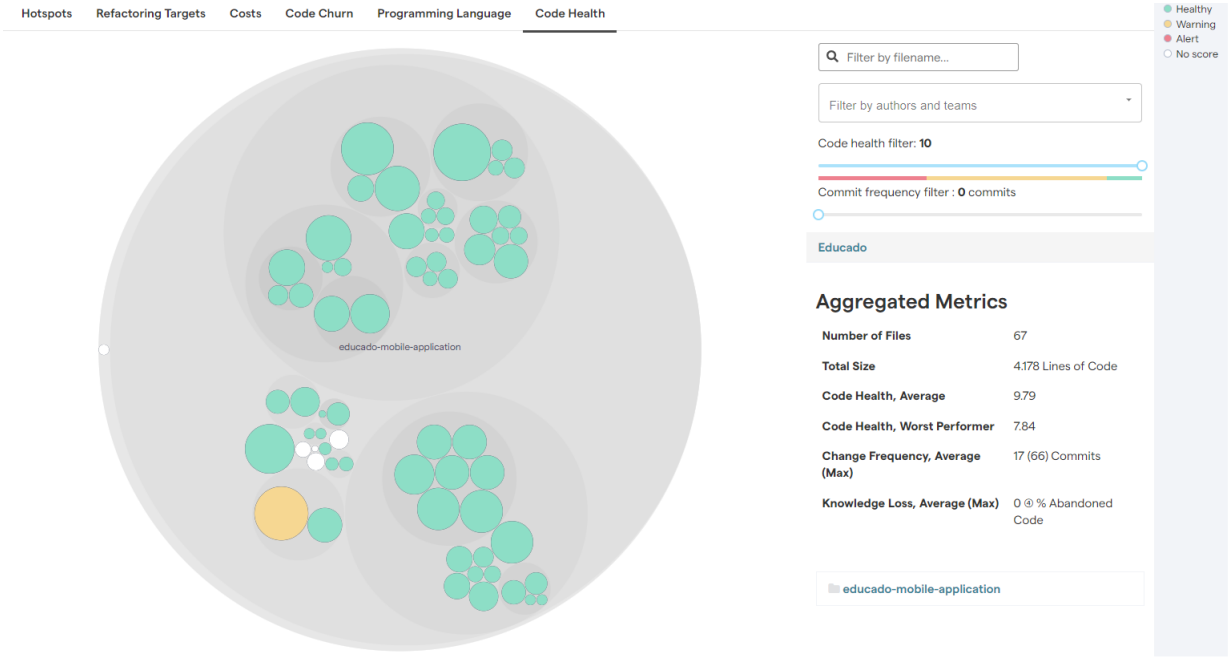


Figure 11.1: CodeScene report of the code health

The figure shows us that we have developed 4.178 lines of code and an average of 9.79 / 10 code health. Whereas the worst performer out of all the files is a back-end-related file, which is not of the group's concern. Nevertheless, this gives us important information about which file could create issues for us in the future. In general, though, we are pleased with the exceptionally high code health.

With CodeScene, we can also check the Code Churn, which means how frequently code has been deleted and rewritten. The figure of this can be seen below:

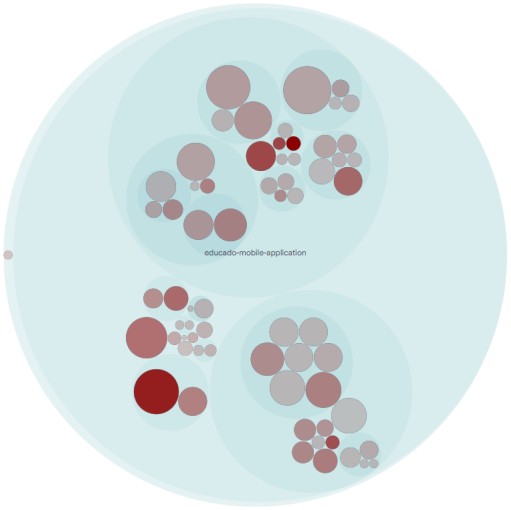


Figure 11.2: CodeScene report of the code churn

With this, we can see which files required a lot of work, which we did not anticipate at the beginning of the project. Many of the files are back-end related but there are few front-end files. Files such as EasyDynamicList.jsx component, which is utilized in the CourseView screen. We can see that it has changed a lot because we were not sure how we should make it dynamically get data from the database. Therefore, the file was using mocked data, to begin with. Once we got a clearer vision of how it should be structured, we rewrote it. This happened a lot of times as we can see above, because its Code Churn is high.

SonarCube

Another tool for static code analysis is SonarCube, which we used to compare the initial state of the code to our final code. Below, an overview of the initial state of the code is provided:

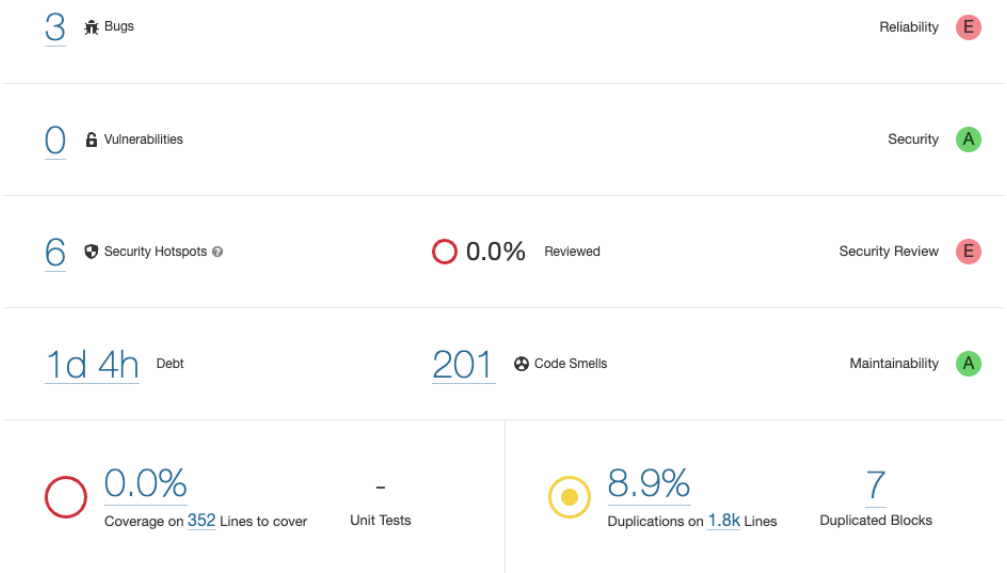


Figure 11.3: Initial state of the code base

The overview shows different interesting statistics, among others the technical debt, the test coverage of the code base, and the amount of code smells, which means code that will be hard to maintain. However it should be noted that SonarCube was not configured to check the coverage properly, so this number is not really meaningful. Nonetheless, the technical debt of the repository was one day and four hours, and over 200 code smells existed when the code was handed over. The following picture shows the result of the SonarCube analysis on our final state of the code base:

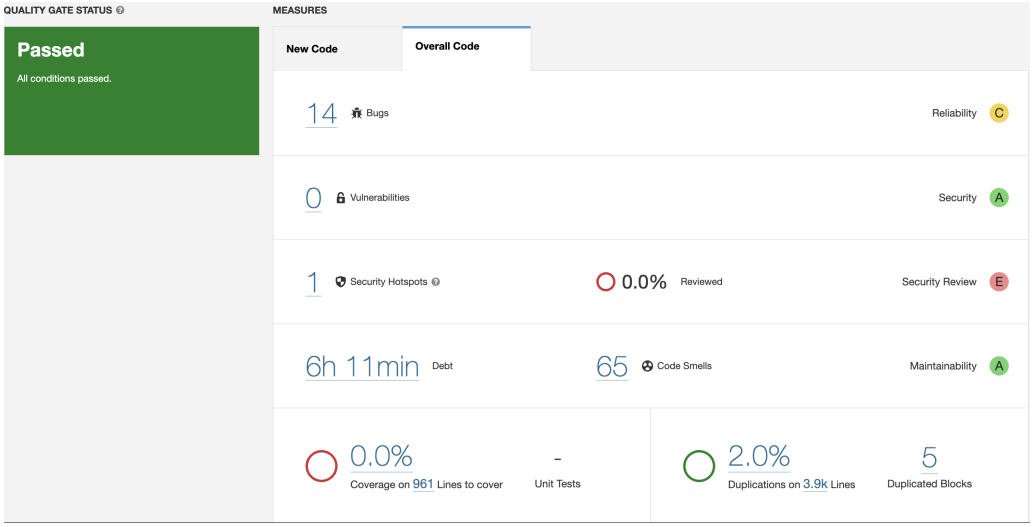


Figure 11.4: Final state of the code base

Here it should be clear that we brought the technical debt down quite a bit to around six hours. Furthermore, the number of code smells was reduced by over half. As a side note, the group found that the number of bugs found by SonarCube can not be trusted when writing code in React Native. A lot of the so-called bugs were implemented hooks that the SonarCube had a hard time understanding the concept of since the hook could change state at runtime.

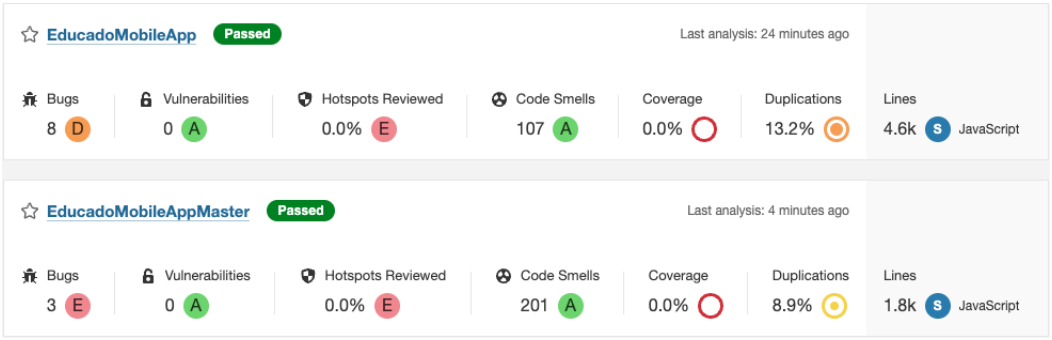


Figure 11.5: Statistic comparing initial state to final state of code

Above, a comparison of the initial code base (EducadoMobileAppMaster) and our final code base (EducadoMobileApp) is provided. Here, the numbers revolving around "code smells" change a bit, although the amount of code smell is still around half in our final version of the code base. This is accomplished even though the amount of lines of code is nearly tripled.

Lastly, a diagram of how the technical debt in the code base is distributed is shown.

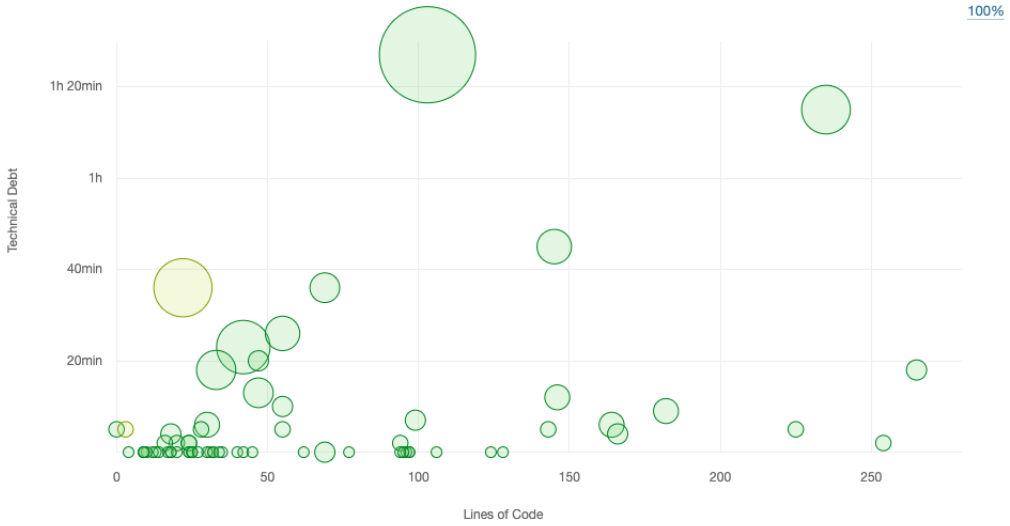


Figure 11.6: Figure of the technical debt in the final state of code

The diagram shows the number of lines of code in relation to the technical debt for each file in the repository. In particular, two large circles stand out, where the largest one is revolving around the "testStack"-file that was used during development. The most interesting file that stands out is the one with the largest amount of lines of code compared to the technical debt. This file is called "StorageService", which acts like the service layers of the mobile side of the Educado application (see figure 4.1). This makes a lot of sense to us, as this was the biggest dependency the group faced throughout the project. We multiple times experienced that we needed something implemented in the service layer in order for the front-end to implement functionality, but the service layer was not ready for that. We tried to communicate our needs to the responsible group, but it ended up being a somewhat slow and agonising process. In the final sprints, however, when the groups started working together more as one team instead of three, the front-end needs were easier implemented as the task was completed through cooperation between team members of both groups.

11.2 Usability testing

During the project, the group discussed multiple times how it could make sense to perform usability testing of the application. One key aspect that caused some frustration was the fact that we did not have any opportunity to get in contact with the real end-users, the Brazilian waste pickers. We could still derive features that the end-user would probably want based on the PO's understanding of the problem domain, however, the actual validation of these was somewhat troublesome to come up with a reasonable way of doing.

The group discussed if performing the usability test on similar users to the waste pickers would be feasible, but the criteria of being of low educational background, with limited ability to read and write limited the field of possible candidates. At one point, elderly people or young children were on the table of possibilities, if it would be possible to get in touch with someone that is struggling with literacy. However, due to several issues, the group eventually decided to hold off on performing the test. Some issues were due to the state of the application which was still being developed into a coherent state by the very last sprint of the project, while other concerns revolved around the fact that even if we got in contact with similar users to the waste pickers, there would still be significant differences. Among others, the fact that the waste pickers know a limited amount of English would mean that the intuitive way of navigating through the application would be different than the test subject that we would be able to perform the testing on.

In the end, the group wanted to perform the usability test, but only in a way that made sense. Because of the reasons listed above, the conclusion ended up being that the test ought to be performed with some of the actual Brazilian waste pickers if the test could be considered to be anything other than performance. Luckily, two of the group members were enrolled in the trip to Brasilia planned by Aalborg University. Therefore, the plan became to have the two group members perform the usability testing while visiting the waste pickers. They will report the findings of the test back to Aalborg University in order for future generations of software students to be able to make use of it.

The following usability test was developed by the group, for the purpose of performing it with waste pickers in Brazil. First off, the test user should be asked some formal questions before the actual testing begin. These include but are not limited to asking if the test may be recorded as well as the test user's name. Afterwards, the Educado application will be installed on the user's phone if possible - alternatively, the user will use a provided Android with the application preinstalled.

Following this, the user will be asked to perform several different tasks, each of which focuses on a specific feature of the application. The tasks will be given as a case, where each case have a setup, something that the user wants to do or achieve in the application, and an open-ended question so that the user has a somewhat free interpretation of what to do. For each task, the group member that will act as an observer takes notes about what behaviour and steps are taken in order to try to complete the given task, whether or not it has been completed, as well as the time to complete it, if successful. An important thing to note is that the following tasks will be done as a think-aloud test, where the user says aloud what he/she thinks at any given time, what their conclusions about the task are, as well as any reasoning for doing what he/she is about to do.

Tasks:	**Observations:**	**Completed:**	**Time to complete:**
1) You want to access the application. What do you do?			
2) You want to have an overview of all the available courses in the application. What do you do?			
3) You want to see the content of a course. What do you do?			
4) You want to try another course. What do you do?			
5) You decide that this course looks interesting. How do you proceed?			
6) You want to start the course. What do you do?			
7) You want to change what part of the course, you are focusing on. What do you do?			
8) You want to get an overview the course you are currently doing. What do you do?			
9) You want to see the information the application has on you. What do you do?			
10) You do not want your account anymore. What do you do?			

Table 11.1: Usability test

When the test is actually being carried out, several obstacles might arise, which we have given some thought to beforehand. A possible issue that could affect the testing is the possible language barrier that might occur between the test user and the group members. It should be noted though, that the PO of Educado, Mateus Halbe Torrés, will be attending the part of the trip to Brazil where the group will interact with the waste pickers. Mateus is Portuguese- as well as English-speaking, so he could step in and act as a translator if needed.

Furthermore, several aspects of the test user might affect the outcome of the testing. The understanding of what it is to be a waste picker is only based on what the PO has told us. At one point he mentioned, that the waste pickers were unlike anyone we have ever met since the closest waste picker in Denmark (a tin deposit gatherer) would still be considered miles apart from the Brazilian one. In the worst-case scenario, the waste pickers could prove unable to perform the testing due to issues that the group would have a hard time predicting, such as the waste pickers' ability to understand the given task, their ability to focus on the task at hand, their ability to verbalise their thought process and so on. We hope to minimize this aspect, by having the waste pickers speak in their native tongue while recording the test, which could then be transcribed afterwards.

Lastly, there are some technical aspects to consider when performing a usability test on the other side of the Earth. For once, the back-end should be deployed on a server closer to Brazil to make the content available there. This will cost money, and therefore this responsibility lies outside of the group grasp.

Additionally, the group members that will travel to Brazil will need a way to get the application onto the waste pickers phone, if possible. Currently, the plan is to have the application installed by entering *developer mode* on their phone, giving access to install the Android Package Kit (APK) via a link to the application stored in an AWS bucket in the cloud. The backup plan could be that the group members who will travel to Brazil will use their own phone and hand it out when the test is ready to be performed, and help when issues might arise. There might also be an option where we bring phones from the department to Brazil in order to have something to perform the test on.

11.3 Snapshot testing

The purpose of a snapshot test is to output a textual form of the rendered component. The textual form is human-readable which makes it easy for us to understand and look for potential bugs. This can be used to ensure the code's quality and test its behaviour. Besides this, we can also use snapshots to compare renders of different iterations of the project.

We were able to only test one of the multiple screens that our application features, this is mostly framework related and we were not able to have it function with everything else that our application offered. To do snapshot testing we utilized a framework called Jest. Before we could even begin with the creation of the test cases we would need to first begin with the configuration of the framework. In our case, the configuration can be found in Appendix in listing A.1.

During the creation of our test cases for the screens, we would need to use multiple hooks and other stuff such as asyncFunctions. To have those functional we needed to mock them using Jest which allows us to mock external dependencies. The mocking can be seen in Appendix in listing A.2

Once we have those mocked we can begin creating our test cases. As mentioned before we could not test all of our screens or components because they contained components that were not compatible with different frameworks. The cases can be seen in Appendix in listing A.3.

The test gives a readable description of how the screen Explore is built. What are the different parameters used to build this screen and show us the different components it is built with. This can be then used to further enhance the understanding of what is happening behind the front-end, down to the code. A snippet of the result can be seen below:

```
/// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`Testing if Explore renders 1`] = `
<View
  style={
    [
      {
        "backgroundColor": "#CFE9EF",
      },
      {
        "flexBasis": "100%",
      },
      {
        "display": "flex",
      },
    ]
  }
>
  <View
    style={
      [
        {
          "flexBasis": "16.666667%",
        },
        {
          "alignItems": "center",
          "justifyContent": "center",
          "paddingTop": 20,
        },
      ]
    }
  >
```

Listing 11.1: Snippet of Jest Test Results

The rest of the snapshot can be seen in Appendix in listing A.4

12 | Discussion

Throughout the project, it became evident that several issues affected the development process and frustration level. In the following, a general discussion about experiences made by the group will be provided after which a section about the future development of the application will be presented.

The group went through several sprints, in the beginning, feeling frustrated about the process that we had to follow when deriving user stories for the upcoming sprint. In the beginning, it felt like there was no common goal for the three teams, which meant that the application as a whole evolved in different directions. For instance, our group had a sprint focusing on gamification, while the other two groups focused on user representation in the different areas of the application. This ended up impacting the possibility of actually implementing the gamification elements that this group produced since it was only the front-end part of the application that was ready for it.

During the project, it also became clear that a common Definition of Done would have been nice to commit to before the project began. Not having one meant that the application as a whole, back-end and front-end, did not end up being as cohesive as it could have been. For instance, a small example could be found in the design of the different screens of the application, where the colour schema for the login screen does not match the rest of the app. The reason for this was that another group was in charge of this screen, but the two groups did not agree upon a common styling beforehand.

When the project concluded, we reflected on what it would take to have the application go into production. Here, it became clear that the teams had developed a lot of technical debt, which they ideally would have liked to fix before deploying. In the following section about future development, the group will make an effort to account for some of the technical debt that relates to the front-end specifically.

Lastly, the group spend quite some time trying to get the local environment up and running, with an emulator that could correctly display the application on a projector when presenting our increment during the sprint review. We started only doing it on one or two laptops, but due to different technical aspects, it ended up not being sufficient. If the group, or future generations of teams, was to start on the future development of the application, it would be strongly advised to get this aspect of the development process done on all team member's laptops from the get-go.

12.1 Future development

Written in collaboration with all groups

In this section, an overview of the most apparent areas for the front-end side of the Educado application will be provided. The areas that will be covered in the section revolve around:

- Gamification
- R2L
- Styling
- Technical debt
- Testing
- React Native
- Possible upcoming features

Gamification

As mentioned earlier in the report as well as stated throughout several sprint goals proposed by the PO, the application should have implemented gamification elements. During Sprint #3, the group even managed to create some of these and, although hard-coded at the time, it was clearly something that both the group as well as the PO would have liked to see implemented. Throughout several sprints, the group would have been able to explore this design space even further, but this was simply not possible because of the fact that gamification related to an actual user could not be identified on the back-end of the application, which was first implemented in the very last sprint of the project. However, the group finds that this would be the first and most vital area to further develop if given more time. Currently only a very slight hint of gamification can be found on the "RightAnswerScreen", which the user sees if he/she answers correctly on an exercise. Here, a star is "given" as a reward together with a text, encouraging the user to continue using the application. The group has taken more elements into consideration, among which a leaderboard should be mentioned. At the concept level, the board should only list the user's known associates that also use the application, in an effort to enable the user to make meaningful comparisons of progression. Furthermore, this indicates that a design space to explore would be to implement a friend list feature in the application so that this relation can be mapped in the back-end. Elements like achievements, and/or overall user progression were also on the minds of the group. These, together with the leaderboard and friend list would in our points of view create the foundation for a waste picker to want to come back and use the application again and again. Lastly, the group wanted to create even stronger user retention in the application by possibly adding notifications that would remind the user to use the app, or inform them that one of their friends just beat their score on the leaderboard or similar events occurring in the application.

R2L

This area revolves around a concept that the group sought to apply through the development of the application. As mentioned earlier in the report, the concept relies on short interactive sessions between the teacher and the learner. Also, the interactions should be based as little on text as possible, which relates to what we currently know about the waste pickers' ability to read and write. Although the group made an effort to keep the amount of text as low as possible, this proved to be difficult at times because of several issues. We did manage to end up using a text-to-speech feature for the different possible answers to the exercises, but since the application did end up with more text than we would have initially liked, this feature could be implemented across all text in the application to further lessen the actual need for reading in the application.

Styling

Throughout the project, the different screens have undergone several design changes. Although our prototypes were made as a starting point, the design changed due to different aspects, one of which was that our understanding of the application and its functionalities evolved over time. In general, the group tried to make a proof-of-concept of the design first, after which the design was iteratively improved upon. During the later stages of the project, the group made an effort to standardize the styling by using the framework tailwind-CSS and chose what we thought would be a nice and easy-to-use library. After working with the library for several sprints, it is the group's assessment that another styling framework ought to be considered implemented instead if given more time. The reasons are plentiful, but one of the bigger ones is the fact that a lot of the library does not work on Android, as the framework has been mostly developed to fit iOS. Another reason is the fact that tailwind-CSS is deprecated and as such, we ended up using a mix of it and its successor called native-wind.

Technical debt

When the group first started working on the code base, it soon became clear that the code base was using deprecated libraries or versions of libraries. This meant that it took several weeks to get the code up and running, and implied using a lot of "hacky" solutions such as, on purpose, rolling back versions of libraries in order for it to work. Throughout the development process of our take on the application, we tried to minimize the number of deprecated libraries, but several still exist. Therefore, this technical debt should be

taken into account if further development on the application should be considered. Furthermore, the group itself produced technical debt when developing. For instance, some implementations were made to work instead of being the correct, clean and efficient way. Ideally, this debt should be sorted, if given the time.

Testing

The group performed several forms of testing, as mentioned earlier in the report. Since our focus is purely front-end based, it made the most sense to implement usability testing compared to other aspects of testing such as unit testing. However, because of the development process between the different groups, the application was only ever really in a state to be tested in the very, very late part of the project. Furthermore, it should be noted that the group did not have access to users that could imitate a Brazilian waste picker in any meaningful capacity. Because of these two reasons, the group chose to hold off on doing the usability testing before the end of the project, even though it was prepared as mentioned earlier in the report. After the time frame of the project has ended, two members of the group will travel to Brazil, where they will have the opportunity to carry out the usability testing with real end-users. Since this has not yet happened, further improvements to the application could become viable after this testing has been done.

React Native

When we got access to the code base, it was written in React Native. Although we, based on recommendations from the former developers, did end up choosing to start the whole development of the front-end again from scratch, the choice of programming language remained the same. While it proved to be doable to write the application in React Native, several issues point to the fact that we maybe should have considered other languages as well before making our decision. For instance, we multiple times experienced that a library was not working because it was discontinued due to lack of maintenance. While we of course can not be sure that the same would not occur if another programming language were used, maybe a more currently chosen language like flutter could have made sense.

Possible upcoming features

As a result of us using an Agile framework while developing our project, we were regularly implementing and brainstorming new features in the form of User Stories. While a fair amount of these features were implemented, many were also dropped for one reason or another. This section will cover the more notable features that were either not finished in time or dropped altogether before work ever began on them.

Adding additional exercise formats
When the group first started off developing, multiple different formats for how an exercise could look were discussed. At that point in time, we went with just a single option, which is the one that is currently implemented in the application. Here, there is a video as a learning input and four possible answers that can be read aloud to the waste picker whenever he/she clicks on a button. However, when learning, not all material will fit this structure well. For instance, the group briefly discussed a format with only two buttons instead of four, or only an audio file instead of a video. There is probably a large design space to explore here, which would make the use of the application more interesting in the long run, as the waster picker will not get bored by the same structure for every single exercise.

Recommender system for courses
In its current state, the application can show all the courses that are available in the back-end database. Furthermore, the group implemented a filter function that can sort the courses into a category such as "Finance". However, although this gives the user the basic functionality of getting to an interesting course easier, this could be assisted if a recommender system was implemented. The complexity of the system could range from very simple, like showing courses that are in the same category as the one the user is currently doing on top of the list, to more complex such as using Pearson correlation between users to find courses that are similar users have liked. This is a somewhat large task, that, if implemented, should be given a lot of time to develop well.

Enabling profile pictures
Another possible feature that could be implemented is to enable the use of the phone's camera while in the application. At first, the use could primarily be to enable a user to take and add a profile picture to their user profile. This would increase the feeling of relating to the application and would be able to enhance the retention that the users will experience.

Adding additional accessibility
From the research that was done three years ago by the former project group[4], the waste pickers have low to no writing and reading skills. Therefore, additional accessibility features should be implemented, if given more time. For instance, the overview of a course ended up with a lot of text, which could prove hard to navigate for the waste pickers. By adding features like text-to-speech across all text in the application, the burden of having to be able to read would lessen.

Translate text in application
To make the design more easily understand the functions there is in the app the text could be translated from English into Portuguese, such that the waste pickers who are able to read Portuguese will be able to read what the buttons do.

Phone Number Authentication
When we were initially implementing our phone number registration system, considerations were made as to whether to add an authentication system to verify that the phone number was valid. This system would work by sending a text message containing a code to the phone number, which would then need to be typed into the Educado app's user interface to verify that the user was in possession of the phone number and had typed it incorrectly. We looked into a few systems that made it possible to send text messages, but all of these required us to set up API keys that would charge a small amount of money each time a message was sent. This meant that we would either have to pay out of our own pockets while testing and implementing this feature or that we would have to request funding from the university, which could potentially be a very time-consuming process. We felt that this feature would not be necessary for what we wanted to accomplish and when discussed with the PO, it was decided that it would be beyond the scope of the project.

Course progression
While course progression was successfully added and stored for each user in the database, this data is never used in the mobile app. Optimally, the user interface would show the user's progress in some way. This could be represented by a progress bar in the course overview or some other way. This feature would give users a better idea of which courses they have completed and which they are in the process of completing, which could incentivize the completion of more courses, leading to a higher level of engagement.

Caching system
One of our initial goals with this project was to implement some sort of caching system that would automatically download and store content from the application locally, such that users would not need to have entire courses to consume at once. This feature ended up being scaled back to simply being able to manually download and delete content, which is a little less user-friendly. A caching system could work by downloading the first few exercises of a course when the user starts that course. This content would be stored locally for some time until an algorithm would determine that the content could safely be deleted. With this feature, users could spend less time worrying about storage space and manually downloading/deleting content, and instead spend more time using the application for its intended purpose.

Notification System
A notification system would make it possible to send notifications to the user's smartphone, such that they will be notified of new content that is now available in the app, and tell them how they have progressed to improve their motivation for completing the courses they have begun participating in.

Application preferences
To better suit, the user's needs and expectations of the app, a set of preferences could be implemented in the application such that the user decides how the application should behave when using it. Examples of preferences could be the opportunity to choose between a light theme and a dark theme, such that they do not feel eye strain if they are using the application in a dark environment. Another example could be to turn notifications off, such that they do not feel bothered by the application throughout the day.

Add animations to front-end
To make the application more user-friendly and fluid, it could feature more animations. Examples of animations could be an animation when the user opens the app, such that they feel the application is responsive and is not frozen. Another example could be more animations when changing between different screens in the app, such as going from the initial login screen to the register screen.

Reward system
To keep the users feeling more motivated to keep using the application and completing courses, a reward system could be implemented. A reward system could include various badges for different achievements, this could be badges for when they have proceeded in a course every day for a week, when they have completed their first course, or when they have completed all courses in a specific category.

Profile screen
Additions to the profile screen could further improve the personalization of the user experience. They could be able to add a biography to their user profile, such that they have a space to tell more about themselves to other users. And they could be able to see their progress on the profile screen, such that it is easy to glance over all the information in the application that is related to them at once. And lastly, they could be able to edit their information such as their phone number, name, and password, in case they have changed since they registered or they made a typo while entering them.

Restricted Routing
Currently, this feature has only been implemented back-end, but the implementation is still missing in the front-end part of the application. It is therefore not possible to add the routing as it will cause the endpoint it is added to, to not function for the application user.

Responsive front end
While using the mobile app on various devices among team members we experienced that the front end would look a bit different on each of them. There were two main issues that were apparent while testing, the first being that the lower button on the login- and register screens would be shifted up a bit once one of the text fields was tapped and the keyboard opened. The second issue was that the two lower buttons on the login- and register screens would have more space between them depending on the device's screen.

End of collaboration

13 | Conclusion

Throughout the report, the group has described how educational content on the application can be delivered to the Brazilian waste pickers by implementing different features like gamification and shaping the content to fit an R2L approach.

The main goal for the project, and the semester overall, was to integrate each group's work into a single cohesive product, so that next year's software students can develop further. We believe as a group, that we have accomplished this. Both the front- and back-end do have a lot of technical debt, as is also described in the discussion (see section 12), but even so, there is a link between the created courses on the web application to the courses shown in the app, which would not be possible if the cohesiveness was not present. The three groups also managed to make use of the Android phone's local storage, so that the waste picker can download a course while having WiFi so that they can consume it later when they might not have internet. This was also a requirement from the PO and therefore important to the entirety of the project.

The toughest part of the project was the goal of having multiple groups work put together into a single product. As it was the first time that any of us had tried not to create every piece of the puzzle, it took all three groups a couple of sprints to get into a reasonable workflow. But once arriving at the last sprints of the semester, the groups were more aligned and were able to work on assignments between groups more efficiently and fluently. Had we found this connection sooner or had tried it before this project, there might have been more things implemented throughout the application, but in the end, the overall main goal was accomplished.

Throughout the different design and implementation phases, the group's main focus was to make a simple, user-friendly application, that the waste picker could easily navigate through, and minimise the number of words present. Hence, we created a smooth design, where the importance of the exercise screen is underlined by the implementation of gamification elements for the answer buttons and a simplistic interface for the video element. To further assist the waste picker, the group implemented text-to-speech on the buttons, so that the user could listen to the answers instead of reading them. As mentioned in the section about the future development of the application (see section 12.1), a lot of interesting features that could be implemented were found, if given additional time. Most of the features revolve around implementing additional gamification elements throughout the entire application, and not just on the exercise screen, as well as lowering the technical debt by cleaning up the code base.

Nonetheless, the group feels like the project's outcome is successful, as we did manage to connect the front- and back-end, while designing a user-friendly application that takes into account the limitations of the user. We believe that the project as a whole has become a better foundation for future development than what we handed over to us at the beginning of the semester.

Bibliography

1. Accessed: 05/12/2022, 2022, (https://www.statista.com/statistics/530481/largest-dump-sites-worldwide/).

2. K. Vasarhelyi, *Environmental Center*, Accessed: 05/12/2022, (https://www.colorado.edu/ecenter/2021/04/15/hidden-damage-landfills) (2021).

3. D. Britze, R. N. Nielsen, "Mobile Education Platform - Smart Caching Learning Materials" (Aalborg University, 2019), (https://projekter.aau.dk/projekter/da/studentthesis/mobile-education-platform(aebca7a9-9b25-4081-89ea-271477091e93).html).

4. D. Britze, J. V. Jensen, "Digital learning platform for waste-pickers in Brazil", Bachelor Thesis (Aalborg University, 2021), (https://projekter.aau.dk/projekter/da/studentthesis/digital-laeringsplatform-for-skraldere-i-brasilien(c4d83ea9-c4f6-4f82-882a-d42fdec142c8).html).

5. *The Agile Manefesto*, 2022, (https://agilemanifesto.org/principles.html).

6. *What is Scrum?*, 2022, (https://www.scrum.org/resources/what-is-scrum/).

7. *[Myth Busting]What is A User Story?*, 2022, (https://www.scrum.org/resources/blog/myth-busting-what-user-story).

8. *User Stories are Needs Described from the Business Perspective*, 2022, (https://www.scrum.org/resources/blog/user-stories-are-needs-described-business-perspectivey).

9. *Scaling Scrum with Nexus*, (2021; https://www.scrum.org/resources/scaling-scrum).

10. F. Ugur-Erdogmus, R. Çakır, (https://journals.sagepub.com/doi/abs/10.1177/07356331211065679) (2022).

11. S. C. Paula Bitrian Isabel Buil, (https://www.sciencedirect.com/science/article/pii/S0148296321002666) (2022).

12. L. F. m. Tong Wang, (https://www.jmir.org/2021/8/e24546/) (2021).

13. D. Rose, Accessed: 13/12/2022, (https://camphillipss.eq.edu.au/curriculum/subjects-and-programs/reading-to-learn-r2-l) (2022).

14. *Dualingo website*, 2022, (https://www.duolingo.com/).

15. (https://en.wikipedia.org/wiki/TikTok).

16. *Expo speech documentation*, Accessed: 20/12/2022, 2022, (https://docs.expo.dev/versions/latest/sdk/speech/).

Part IV

Appendix

A Testing

A.1 Snapshot Code snippets

```
//** @type {import('ts-jest').JestConfigWithTsJest} */
module.exports = {
  preset: 'jest-expo',
  testEnvironment: 'node',
  transform: {
    '^.+\\.ts?$': 'ts-jest',
    "^.+\\.(js|jsx)$": "babel-jest",
  }, moduleFileExtensions: [
    'ts',
    'tsx',
    'js',
    'jsx',
    'json',
    'node',
  ],
  setupFiles: [
    "./jestSetup.js"
  ],
  transformIgnorePatterns: ["node_modules/?!(@expo-google-fonts|expo_font)"],
  globals: {
    "__DEV__": true
  },
  moduleNameMapper: {
    "expo-font": require.resolve('expo-font'),
  }
};
```

Listing A.1: Jest Config

```
/import React from "react"
import { render, screen } from "@testing-library/react-native"
import Explore from "../../screens/explore/Explore"
import { expect, test } from "@jest/globals"
import { useNavigation } from "@react-navigation/native"
import { useRoute } from "@react-navigation/native"

import { Animated } from 'react-native';
jest.mock('@react-native-async-storage/async-storage', () =>
    require('@react-native-async-storage/async-storage/jest/async-storage-mock')
);
jest.mock('react-native-reanimated', () => {
    const Reanimated = require('react-native-reanimated/mock');
    Reanimated.default.call = () => { };

    return Reanimated;
});
jest.mock('react-native/Libraries/Animated/NativeAnimatedHelper');
const mockedUsedNavigate = jest.fn();
const mockedUseRoute = jest.fn();
jest.mock('@react-navigation/native', () => ({
    ...jest.requireActual('@react-navigation/native'),
    useNavigation: () => mockedUsedNavigate,
    useRoute: () => mockedUseRoute,
    useParams: () => ({
        courseId: 0,
        sectionId: 0,
    }),
}))
const mockPlatform = OS => {
    jest.resetModules();
    jest.doMock("react-native/Libraries/Utilities/Platform", () => ({ OS, select: objs =>
        objs[OS] }));
};
```

```
34 jest.useRealTimers()
```

Listing A.2: Jest Mocks

```
1 /it("my test on Android", () => {
2     mockPlatform("android");
3 });
4 test('Testing if Explore renders', () => {
5     render(<Explore />);
6     expect(screen.toJSON()).toMatchSnapshot();
7 });
```

Listing A.3: Jest Test Cases

```
1  /// Jest Snapshot v1, https://goo.gl/fbAQLP
2
3  exports[`Testing if Explore renders 1`] = `
4  <View
5    style={
6      [
7        {
8          "backgroundColor": "#CFE9EF",
9        },
10       {
11         "flexBasis": "100%",
12       },
13       {
14         "display": "flex",
15       },
16     ]
17   }
18 >
19   <View
20     style={
21       [
22         {
23           "flexBasis": "16.666667%",
24         },
25         {
26           "alignItems": "center",
27           "justifyContent": "center",
28           "paddingTop": 20,
29         },
30       ]
31     }
32   >
33     <Text
34       style={
35         [
36           {
37             "color": "#4b5563",
38           },
39           {
40             "fontFamily": "VarelaRound_400Regular",
41             "fontSize": 40,
42           },
43         ]
44       }
45     >
46       Educado
47     </Text>
48   </View>
49   <View
50     style={
51       [
52         {
53           "width": "91.666667%",
54         },
55         {
56           "alignSelf": "center",
```

```
          },
          {
            "paddingBottom": 16,
          },
          {
            "elevation": 15,
            "zIndex": 15,
          },
        ]
      }
    >
      <View>
        <View
          accessible={true}
          collapsable={false}
          focusable={true}
          onClick={[Function]}
          onResponderGrant={[Function]}
          onResponderMove={[Function]}
          onResponderRelease={[Function]}
          onResponderTerminate={[Function]}
          onResponderTerminationRequest={[Function]}
          onStartShouldSetResponder={[Function]}
          style={
            {
              "borderColor": "gray",
              "borderRadius": 10,
              "borderWidth": 1,
              "flexDirection": "row",
              "justifyContent": "space-between",
              "opacity": 1,
              "paddingHorizontal": 20,
              "paddingVertical": 12,
            }
          }
        >
          <Text
            style={
              [
                {
                  "fontFamily": undefined,
                },
                undefined,
              ]
            }
          >
            Filter by
          </Text>
          <Image
            resizeMode="contain"
            source={1}
            style={
              {
                "height": 20,
                "width": 20,
              }
            }
          />
        </View>
      </View>
    </View>
    <RCTScrollView>
      <View>
        <View
          style={
            [
              {
                "flexWrap": "wrap",
              },
              {
```

```
127                 "flexDirection": "row",
128               },
129               {
130                 "flexBasis": "0%",
131                 "flexGrow": 1,
132                 "flexShrink": 1,
133               },
134               {
135                 "justifyContent": "space-evenly",
136               },
137             ]
138           }
139         />
140       </View>
141     </RCTScrollView>
142   </View>
143   `;
```

Listing A.4: Jest Test Results