# DSSMV 2025/2026 (ProjectReact) - Challengers

Miguel Silva and Lucas Santos

Instituto Superior de Engenharia do Porto,
R. Dr. António Bernardino de Almeida 431, 4249-015 Porto
`{1241131,1241008}@isep.ipp.pt`
`https://www.portal.isep.ipp.pt`

**Abstract.** This report describes the development journey of Challengers. It features the app's context, diagrams, code snippets, functional tests, and potential improvements. Challengers is a multiplayer trivia game, designed for maximizing general knowledge while having fun with friends.

**Key words:** React Native, Typescript, Expo, Expo Go, Challengers, i18n, Supabase, Supabase Realtime, restDB, Open Trivia DB, Game, Multiplayer, Flux, MMKV, Trivia, ISEP, IPP, DSSMV

# Table of Contents

## List of Figures

## List of Tables

# 1 Introduction

Challengers is a multiplayer trivia game for Android, iOS, and web, designed to help you relax from the hustle and bustle of everyday life while learning general knowledge with friends. For most people, everyday life is tough. Lots of work, little rest. When you do rest, you do unproductive things, wasting time. With Challengers, users can have the same level of relaxation and rest while playing and learning with friends, or on their own. The application, developed for all mobile-accessible platforms, was written in React Native, using the Expo framework and TypeScript for more type-safe writing.

## 1.1 Features

The app provides features that embrace the user experience to another level. The following list demonstrates a panoply, but only some examples, of the app's features:

- Ability for the user to customize their app experience:
  - Profile picture
  - Username
  - App theme (light/dark)
  - App language (English, Portuguese, or Spanish)
- Global and user-specific leaderboard
- Customizable game room creation
  - Number of players (1 - 8)
  - Number of questions (1 - 50)
  - Question category
  - Question difficulty
  - Question type
- Join a room shared by another user
- Shareable game room via URL
- Shareable game results via image
- Real-time multiplayer gameplay with synchronized question transitions and answer submission
- Real-time player presence tracking and synchronized game start
- Token-based Authentication
- Export user data and game history as JSON

# 2 Requirements Engineering

The present chapter details the process of Requirements Engineering for Challengers. The main objective during this phase was to understand how to make the app engaging to the user, in solid requirements. Firstly it will be presented the Functional Requirements, expressing the app's capabilities. Then, the Non-Functional Requirements, emphasizing in performance, reliability and usability. Lastly, the app's behavior will be modeled by the Use Case Diagram, showcasing the direct interactions between the user and the app.

## 2.1 Functional Requirements

The system must allow the user...

– To change his/her information
– Create and join a game room with personalized parameters
– To change the app language/theme

## 2.2 Non-Functional Requirements

### Usability

– Graphical interface.
– Optimal user experience (UX).
– Light and dark mode availability.

### Reliability

– Data Storage must be reliable.

### Performance

– Battery consumption must be minimal.
– App must boot in under 6 to 7 seconds.

### Supportability

– Full code and builds available on the official GitHub page.
– Well-documented technical report.
– Further documentation available on: https://deepwiki.com/8126Lucas/DSSMV_ProjectReact_1241131_1241008.

### Design Constraints

– Data persistence and game mechanics require an internet connection (HTTP/WSS requests).

### Implementation Constraints

– Typescript programming language and React Native with Expo framework.
– MMKV data persistence package.
– Unsigned iOS build file (IPA).
– API's free-trial limits

### Interface Constraints

– React Native components.

### 2.3 Use Case Diagram

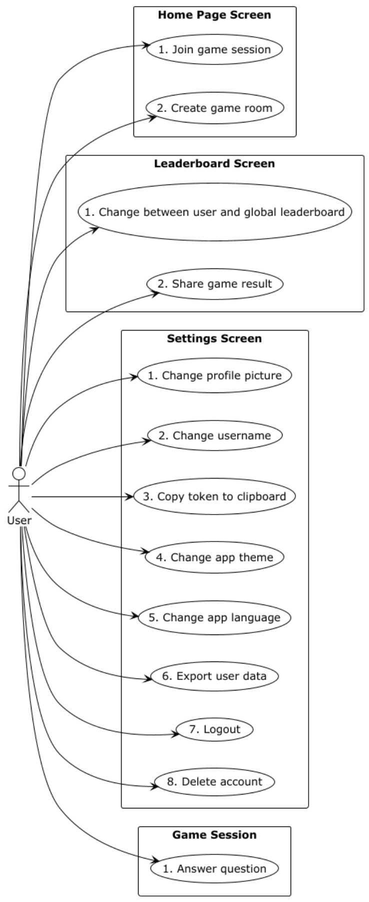


Fig. 1: Use Case Diagram

## 3 Architectural Design

Challengers can be considered a complex system. Between real-time game organization, multiple players, and post-game statistics accessible to all users, it is necessary to develop an excellent architectural design, making all functionalities amenable to improvement, but not modification, and also ensuring that anyone in the community can easily access and utilize the source code.

To be successful, we present a Domain Model representing the most important business classes, most of which are not promoted to software classes; a

Component Diagram defining the internal and external connections of the application, showing the complete Challengers system apart from the user's device; and a Package Diagram revealing the structure of the source code, contributing to the community's knowledge.

### 3.1 Domain Model

Our Domain Model represents a simple yet logical structure of challengers.

As a primary step for development, only a few concepts were selected to become software classes. *TriviaObject* and *GameScore* are now software classes. *Room*, *PlayerScore*, *GameScore*, *GameScoreMetadata* and *User* refer to data stored in the Supabase Database and restDB, which are later uploaded to Challengers.

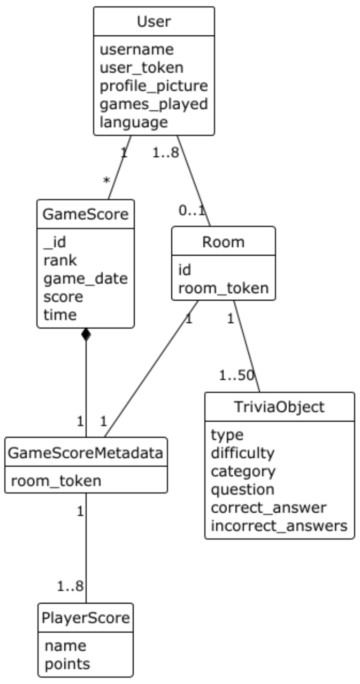


Fig. 2: Domain Model

### 3.2 Component Diagram

As of now we must demonstrate how our components are connected, therefore this diagram reflects on how Challengers (client), as an app, communicates to it's, somewhat large, stack of REST APIs.

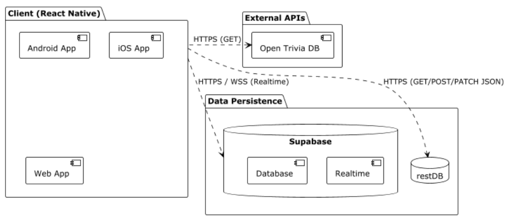


Fig. 3: Component Diagram

### 3.3 Package Diagram

In favor of project maintenance and comprehension, it is important to layout Challengers' organization.

All screens are inside the "app" package, and custom components used in said screens are inside the "components" package.

Inside the "src" package is a wide range of materials designed by us to improve the overall user experience (UX). The "i18n" package contains the dictionaries that facilitate internationalization through Challengers. The Flux architecture, implemented in Challengers with Redux, is specified in the "flux" package. Lastly, designed as a special bonus to maximize UX, the "foreign" package involves a script, with the developer running a Docker image of LibreTranslate to translate all questions available in the Open Trivia DB's database to a Supabase table.

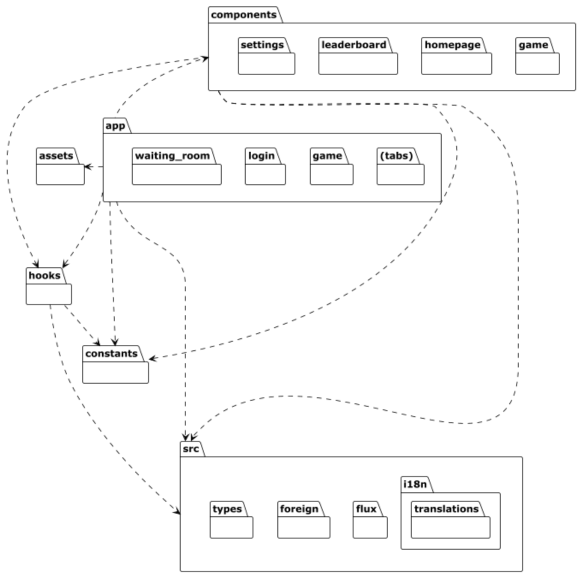
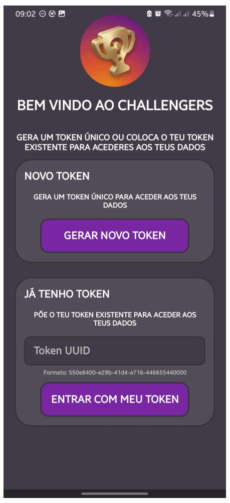


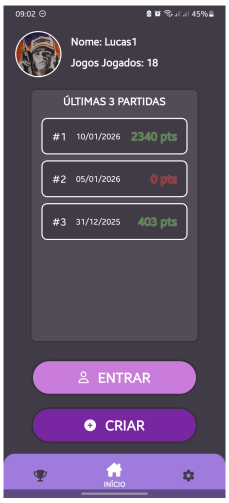
Fig. 4: Package Diagram

### 3.4 User Interface

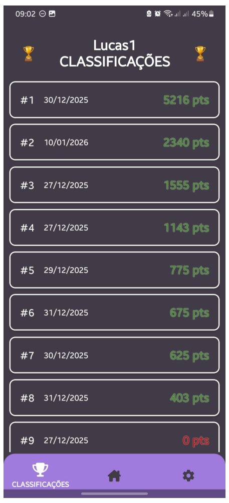
This section presents the thiking behind Challengers' UI. For the color palette, we felt it was best to use colors that conveyed what Challengers represents: knowledge, fun, competition, and luxury. In light mode, we used a mix of warm colors: orange, red, beige, and cream. In dark mode, we tried something different, using more contrasting and sophisticated colors, such as black, violet, and pink. The following images show the main screens in dark mode.


BEM VINDO AO CHALLENGERS
GERA UM TOKEN ÚNICO OU COLOCA O TEU TOKEN EXISTENTE PARA ACEDERES AOS TEUS DADOS
NOVO TOKEN
GERA UM TOKEN ÚNICO PARA ACEDER AOS TEUS DADOS
GERAR NOVO TOKEN
JÁ TENHO TOKEN
PÕE O TEU TOKEN EXISTENTE PARA ACEDER AOS TEUS DADOS
Token UUID
ENTRAR COM MEU TOKEN

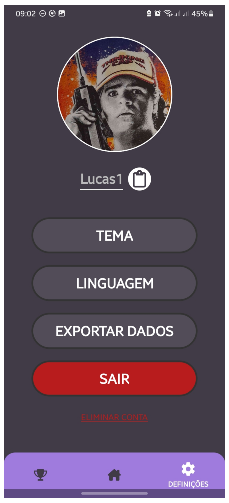
(a) Login Screen



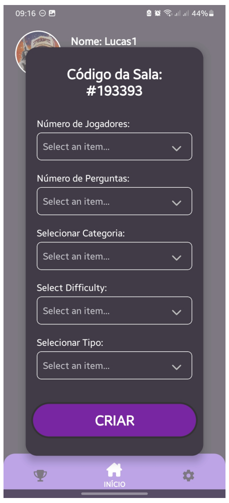
(b) Homepage



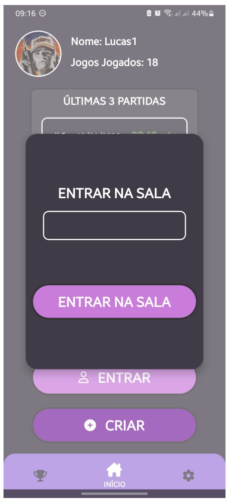
(c) Leaderboard



(d) Settings



(e) Create Room Overlay



(f) Join Room Overlay

Fig. 5: Screenshots of Challengers' UI

# 4 Behavioral Modeling

In this chapter we'll demonstrate how a game session behaves during a question. Therefore, to that end, we show the System Sequence Diagram and the Sequence Diagram for the first, and only, Use Case for the Game Session.

## 4.1 System Sequence Diagram

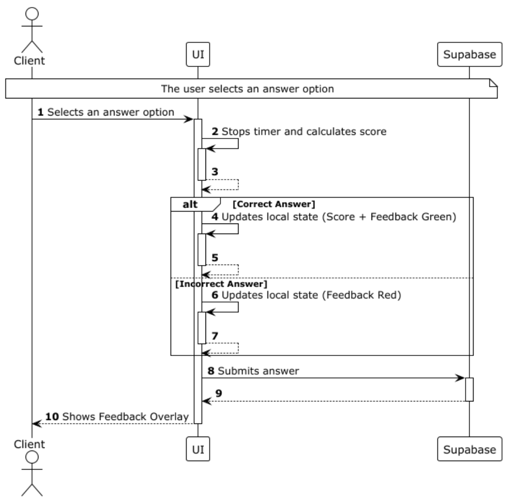

Client
UI
Supabase
The user selects an answer option
1 Selects an answer option
2 Stops timer and calculates score
3
alt
[Correct Answer]
4 Updates local state (Score + Feedback Green)
5
[Incorrect Answer]
6 Updates local state (Feedback Red)
7
8 Submits answer
9
10 Shows Feedback Overlay
Client
UI
Supabase

Fig. 6: Game Session's UC1 System Sequence Diagram

## 4.2 Sequence Diagram

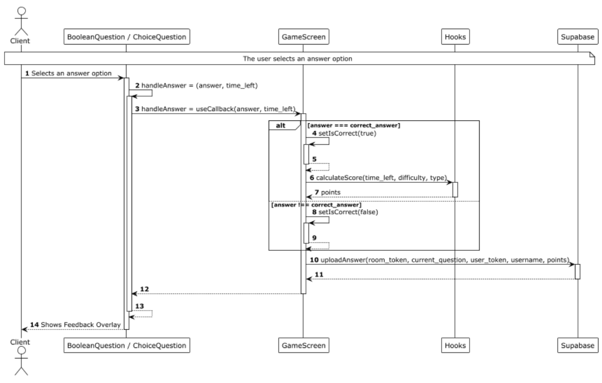

Client
BooleanQuestion / ChoiceQuestion
GameScreen
Hooks
Supabase
The user selects an answer option
1 Selects an answer option
2 handleAnswer = (answer, time_left)
3 handleAnswer = useCallback(answer, time_left)
alt
[answer === correct_answer]
4 setIsCorrect(true)
5
6 calculateScore(time_left, difficulty, type)
7 points
[answer !== correct_answer]
8 setIsCorrect(false)
9
10 uploadAnswer(room_token, current_question, user_token, username, points)
11
12
13
14 Shows Feedback Overlay
Client
BooleanQuestion / ChoiceQuestion
GameScreen
Hooks
Supabase

Fig. 7: Game Session's UC1 Sequence Diagram

## 5 Implementation

Still regarding the Use Case previously mentioned, it's implementation required a complex game logic, with multiplayer synchronization, translation, and time-based point calculation features.

When a guest answers, a row of data is sent to a Supabase table, in our case "in_game_answer_state" acknowledging the guest's answers. When the host has answered and checks that everyone else has also answered, it sends a broadcast to every guest, notifying them to go to the next question.

| **Technology** | **Version** | **Purpose** |
|---|---|---|
| **Expo SDK** | ~54.0.25 | Framework layer providing native module access and build tools |
| **React Native** | 0.81.5 | Cross-platform UI framework |
| **expo-router** | ~6.0.15 | File-based routing system |
| **Redux Toolkit** | ^2.11.1 | Global state management |
| **Supabase JS** | ^2.86.0 | Real-time synchronization (WebSocket channels, presence, postgres_changes) |
| **i18next** | ^25.7.3 | Internationalization framework |
| **TypeScript** | ~5.9.2 | Static type checking |
| **react-native-mmkv** | ^4.1.0 | High-performance local storage |
| **expo-secure-store** | ~15.0.7 | Encrypted credential storage |

Table 1: Technology Stack

One of the reasons why the Expo framework was chosen to develop this application was Expo Go. However, with the continuous integration of features and new technologies (seen in table 1) in Challengers, we saw the need to improve the performance of loading and writing data locally. To this end, we replaced AsyncStorage with MMKV, which is advertised as being up to thirty times faster. Since MMKV uses C++ modules, we had to move away from using Expo Go, as it only supports bundles of modules written in JavaScript/TypeScript, and we started using development builds.

With automation in mind, we created CI/CD pipelines to facilitate the publication of new versions of Challengers. Currently, there are workflows using GitHub Actions for web deployment, iOS build, and Android build. Referring to the iOS version of Challengers, it should be noted that, as the project is intended for higher education, an Apple developer account was not used; therefore, for the application to function correctly, AltStore must be used with its limitations. On Android, you simply install the APK as usual.

## 6 Testing

Throughout this chapter we will demonstrate some functional tests that any user can perform while using Challengers.

These tests were performed using a Samgung Galaxy A12 (Android 13), iPhone 13 Pro (iOS 26.3), OPPO A15 (Android 10), Xiaomi Redmi Note 11 (Android 13), Xiaomi Redmi 13 Pro Plus (Android 15) and the Google Chrome browser.

| **Test** | **Status** | **Flow** |
|---|---|---|
| **Change username** | ✓ | Settings Screen -> Username placeholder |
| **Change profile picture** | ✓ | Settings Screen -> Profile Picture Placeholder OR Homepage Screen -> Profile Picture Placeholder |
| **Change app's theme** | ✓ | Settings Screen -> App's Theme Button |
| **Export user's data** | ✓ | Settings Screen -> Export Button |
| **Logout** | ✓ | Settings Screen -> Logout Button |
| **Delete account** | ✓ | Settings Screen -> Delete Account Text |
| **Change between user and global leaderboard** | ✓ | Leaderboard Screen -> Leaderboard Title Text |
| **Share game metadata** | ✓ | Leaderboard Screen -> Game row -> Share Button |
| **Join game room** | ✓ | Homepage Screen -> Join Button -> Insert Code -> Join Button |
| **Create game room** | ✓ | Homepage Screen -> Create Button -> Insert Game Data -> Create Button |
| **Answer question** | ✓ | Game Screen -> Answer Question -> Points Overlay -> Next Question |

Table 2: Functional Tests

## 7 Conclusion

To conclude the Challengers presentation, we must emphasize that the application is better than our initial expectations, due to the hard work and guidance received from the teachers.

However, it should be noted that there were opportunities for improvement, such as seeking alternative APIs to overcome the limitations of the free plans and improving the context of translations, improving the UI, making it more interesting and visually appealing, and finding a better alternative to generate the iOS build without needing AltStore, reducing some friction for the end user.

## A Appendix A: Detailed Use Case Specifications

To see more SDs and SSDs, please follow the link to our GitHub page: https://github.com/8126Lucas/DSSMV_ProjectReact_1241131_1241008/tree/master/UMLs/puml