Universidade de Aveiro
2025

Pedro
Azevedo

TÍTULO DA TESE (MÁXIMO 130 CARACTERES)

The Playground: mobile platform to connect amateur basketball players

Universidade de Aveiro
2025

Pedro Azevedo

TÍTULO DA TESE (MÁXIMO 130 CARACTERES)

The Playground: mobile platform to connect amateur basketball players

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Ilídio Oliveira, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

agradecimentos / **acknowledgements**

Agradeço toda a ajuda a todos os meus colegas e companheiros.

palavras-chave

texto livro, arquitetura, história, construção, materiais de construção, saber tradicional.

resumo

Um resumo é um pequeno apanhado de um trabalho mais longo (como uma tese, dissertação ou trabalho de pesquisa). O resumo relata de forma concisa os objetivos e resultados da sua pesquisa, para que os leitores saibam exatamente o que se aborda no seu documento.
Embora a estrutura possa variar um pouco dependendo da sua área de estudo, o seu resumo deve descrever o propósito do seu trabalho, os métodos que você usou e as conclusões a que chegou.
Uma maneira comum de estruturar um resumo é usar a estrutura IMRaD. Isso significa:

- Introdução
- Métodos
- Resultados
- Discussão

Veja mais pormenores aqui:
`https://www.scribbr.com/dissertation/abstract/`

keywords textbook, architecture, history, construction, construction materials, traditional knowledge.

abstract

An abstract is a short summary of a longer work (such as a thesis, dissertation or research paper).
The abstract concisely reports the aims and outcomes of your research, so that readers know exactly what your paper is about.
Although the structure may vary slightly depending on your discipline, your abstract should describe the purpose of your work, the methods you've used, and the conclusions you've drawn.
One common way to structure your abstract is to use the IMRaD structure. This stands for:

- Introduction
- Methods
- Results
- Discussion

Check for more details here:
`https://www.scribbr.com/dissertation/abstract/`

Contents

List of Figures

List of Tables

Glossário

CHAPTER 1

Introduction

CHAPTER 2

Motivation And Problem Context

2.1 Basketball Context in Portugal

Portugal is clearly not known for its basketball, the quality is far behind other European countries. In the International Basketball Federation (FIBA) world ranking[1], Portugal is in the 47 place, and in the middle of European table in 25th place. Comparing with our neighbours Spain that is in 7th in the world ranking. However, in the past years, the Portuguese basketball achieve some marks.

First, Portugal had its first player in one of the best women league in the world, “Ticha” Penicheiro played during 15 seasons in the WNBA in th USA, winning a title with the Sacramento Monarchs, and some individual awards. And in 2019, entered the Women’s Basketball Hall of Fame, that has the goal of honouring the most influential players in women basketball.[2]

After her, Neemias Queta was drafted to the NBA in 2021, being the first Portuguese in the NBA, and in the 2023–24 season won his first title with the Boston Celtics. Besides not being the most valuable player in his team in the USA, he played an important role with the National team in second appearance in the EuroBasket tournament in 2025. In this campaign, the Portuguese team made a surprise performance, passing through the group phase and then, confronting the German team. In this game the Portuguese team was holding up until the last quarter, where the world champions and the winners of the EuroBasket 2025 took the victory. After the first victory in the EuroBasket, Neemias mentioned that it was a great moment fo the sport in Portugal and wants the sport to grow more in Portugal.[3] To help with this goal, a survey was made in order to understand how players and basketball enthusiasts use the public basketball courts, the frequency, and the problems.

2.2 Problem Statement

After a some conversations with few basketball players and leaning heavily in a survey responses, basketball players and enthusiasts wish to go more often to public courts. Besides time constraints, this two scenarios and the following complains are in these players mind:

Key Scenarios.

- **Insufficient players**: people arrive to play but cannot form a team because not enough players are present;
- **Overcrowding**: courts are full, forcing players to wait or leave.

Common annoyances reported.

- Poor or unpredictable court conditions;
- Courts that are either empty or overcrowded;
- Social problems between players, including aggressive or unpleasant behaviour, individualistic style, and mismatched competitiveness (players who do not take the game seriously or whose skill and attitude creates a poor experience).

These issues point to a coordination and information problem. Basketball players lack a platform to discover, schedule and evaluate informal games, and to verify court availability and condition. As a result, players often waste time visiting courts with unsatisfiable condition, which decreases participation and lower quality of both social and sporting experience.

2.3 Proposed Solution

The proposed solution is to design and implement a digital platform that addresses the coordination and information gaps in amateur basketball in public courts, by focusing on two primary features:

Informal Games "Pickup games"[1] - Enable players to create, discover and join informal matches (pickup games), with features that support different play modes and social organization:

- Players can create, search for and join games listed on the platform;
- The games can be competitive or casual;
- Competitive games contribute to a leaderboard/raking table per court;
- Game formats include 1v1, 3v3, 4v4, 5v5;
- Users can create and mange persistent teams and challenge other teams;
- Users can set a team as "next"[2] to a game, to play against the winner.

Court Availability and condition awareness Provide users with real-time and information about courts so they can decide where and when to play:

- Persistent court catalogue with attributes - name, location, full/half court, number of courts, has water fountain, etc;
- Users can indicate intent to attend a court at a specific time, so others can have an idea of the availability;

[1]"Pickup games" is the name used by basketball players to call a game without a formal organization (not in a league, no referee, no strick rules)

[2]"next" is used in the casual pickup games to tell the current teams playing, that wants to play against the winning team.

- User can mark themselves “present” at a court;
- Optional automatic geolocation-based check-in to simplify presence signalling;
- Live occupancy indicators to show how many users intend to go and how many are currently present;
- Court condition reporting by the users, they can submit short status updates to report issues.
- Display short weather summary for the court location (optional)

CHAPTER 3

State of the Art & Related Work

3.1 Mobile Application Development Approaches

This platform has to reach the maximum basketball players for it to work properly, and should work when the players are in the court, or anywhere. In order to achieve this, the platform should be access from the mobile phones, and Statcounter[1] Global Stats reports the following about the Mobile Operating System Market Share in Portugal in September 2025: 65.65% is from Android, and 33.94% is iOS[4]. Despite the fact that more than half of people uses Android, the goal is still reach the maximum players possible, so the platform should be accessible from Android and iOS phones.

In order to develop a mobile application for both platforms, there are different options of approaches that can be selected for the implementation, which are the native, cross-platform, web, hybrid, modeling, cloud-based and merged approaches[5].

3.1.1 Native Apps

Going Native, means that the app will be developed with to a specific OS, using its programming language. To build native Android applications, developers typically use Kotlin or Java. Both language are compiled to Java Bytecode that runs on the Android Runtime (ART). As for iOS, developers use Swift or Objective-C, mainly the first one, to build native iOS applications.

With native apps, there is no need for third-party frameworks, as the code is compiled directly into the platform's native language. This results in better performance and faster execution compare to other solutions, which will be discussed later. In addition to performance, native development offers security benefits, since developers can take advantage of the built-in security features of the operating system. When it comes to User Interface (UI) and User Experience (UX), each platform provides its own guidelines: Android follows Material Design, while iOS follows Human Interface guidelines. This consistency makes apps easier to navigate, and there is

[1]Statcounter is a company that provides a tool for users to get insights of their website's visitors

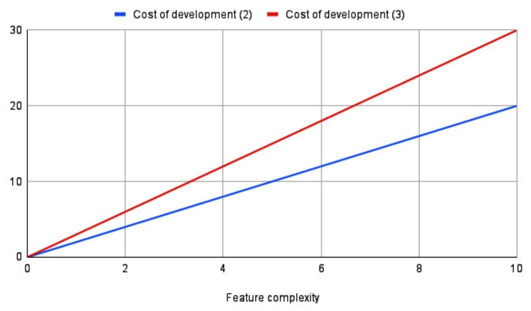
no issues in the app adaptation to other devices. Finally, native applications can make full use of device features such as the camera, microphone and GPS, which enhances functionality and integration.[6]

When the goal is to deliver an application for both Android and iOS using native development, each feature must be implemented twice, once for each platform and in different programming languages. As Robert Nagy explains in *Simplifying Application Development with Kotlin Multiplatform Mobile*[6], the development cost of this approach can be expressed as:

$$Costofdevelopment(n) = n * FC$$

where n represents the number of platforms and FC the *feature complexity*, defined as the sum of all sub-features that compose a feature. The Figure 3.1 , taken from Robert Nagy' book[6], illustrates how the development cost grows for two platforms (blue line) and three platforms (red line) as the number of features increases, assuming no cost reduction from sharing code.

Figure 3.1: Cost of native development as a function of feature comlpexity [6]



Beyond this calculation, additional factors complicate native development. Each platform has distinct characteristics, and a solution that is straightforward to implement on one platform may be unavailable or significantly harder to achieve on another. This divergence creates extra overhead in aligning features sets across platforms. So, Nagy introduces the additional component, *Synchronization Cost*, which increases in an exponential way as the number of features and its complexity grows. The updated formula is expressed as:

$$Costofdevelopment(n) = n * FC + SyncCost^{FC}$$

3.1.2 Web Applications

Any application that is access through *HTTP* requests over a network is called a web based application. These applications are implemented with HTML, CSS and JavaScript. Until Progressive Web App appeared, these applications were browser only, but now, they can be installed in the device, can have offline availability, push notifications and background sync, also able to run on desktop. Developers normally use Angular or React, and other frameworks are

StencilJS or Svelte. However, PWAs cannot access key platform's features as the Native apps can, and also, has a possible higher energy consumption.[7]

3.1.3 Hybrid Applications

Another option, also leveraging web technologies such as HTML, CSS and JavaScript, is taking a hybrid approach. The app is developed with as web app and developers wrap the app via cross-platform wrappers, frameworks and tools, as a native app. With this approach, the app as a user experience just as a native app. Some hybrid app frameworks are Cordova, Phonegap, Titanium, Ionic, being the first the most common. Using third-party plugins with JavaScript, it is possible to access features form the device, such as the camera and GPS. In the Maher Gerges and Ahmed Elgalb article[8], where they compare different approaches, it is compared the hybrid apps directly with the native apps, and, in CPU and Memory occupancy, the hybrid cost more, being 106% higher and 73.0% higher in order. Also, not all features are available and the performance is worse than native apps.[8]

3.1.4 Cross-Platform Apps

In order to avoid repeating the same code for different platforms, there are Cross-Platforms approaches. With this method, it is possible to create applications that run on multiple operating systems. In a first look, using cross-platform methods can be an effective solution to develop applications for a multiple platforms, when time, cost and basic interoperability is the primary concern. On the other hand, Cross-Platform applications tend to have the worst performance and user experience compared with Native apps, as they are behind the evolution of Android and iOS and cannot take full advantage of each platform's features[6]. In different articles and papers, the categorization of the approach varies, especially, when it comes to the Cross-Platform approach, there are subcategories[9]. The Web and Hybrid approach is normally inserted in this category, alongside with Interpreted approach and Cross-Compiled approach[9], which the most popular used frameworks are React Native and Flutter, respectively.

Interpreted Approach - React Native

Similar to the web and hybrid approach, uses web technologies such as CSS and JavaScript, however, the user interface is not HTML and not render in the browser.[7] The most popular used framework is React Native from 2015, developed by Meta, where developers build applications with React components, and the logic in JavaScript. This framework uses component-based architecture, following the steps of React for the web, promoting code reusability, modularity and easy debug.[10] The JavaScript code executes on a separate thread and communicates with the native layer through a bridge[6], allowing developers performance-critical parts of the app in native languages[10]. The bridge translates asynchronous, serializable data between the two environments, which enables native rendering of components. However, this approach introduces overhead, resulting in a process generally slower compared to fully native applications.[6]

Cross-Compiled Approach - Flutter

In this approach, the app is developed in a common language, and then, it is compiled to the native code that can be executed on a mobile device, with Flutter and Xamarin popular examples.[7] Flutter is a framework developed by Google, its architecture can be described in three layers, as noted by Roby Nagy in his book[6]. The Framework layer, where developers write the application code and the UI components, declaratively using the Dart language. The Engine layer renders Flutter widgets to a canvas called Skia Canvas, which is then passed to the final layer, the native platform. The platform displays the canvas and sends back user events to the framework.[6] Compared to React Native, Flutter often achieves better performance by relying on Android's Native Development Kit and iOS's Low-Level Virtual Machine to compile code coming from the engine, Flutter applications deliver higher execution performance. Nevertheless, certain parts of a Flutter application may still need to be implemented in Java/Kotlin and Obj-C/Swift code. Communication between these native components and the Dart code is through a channel, reducing scalability[6].

Cloud-based Approach and Modelling Approach

A Cloud-based application, has its backend in the cloud, instead of running locally, using the cloud features such as flexibility, virtualization and security.[9] Companies like Amazon Web Services, offer that service called Backend as a Service (BaaS).[5]

There are two subcategories in this approach, the Model-Based User Interface Development (MB-UID) and the Model-Driven Development (MDD). Developers use abstract model to describe tasks, data, users and the user interface, which will generate source code. It can transform to code from different platforms. Some tools include XMobile, for MB-UID and JavaScript Application Framework for MDD.

3.1.5 Kotlin Multiplatform

Although cross-platforms the previous frameworks solve the problem of dealing with different platforms, it is still not optimal if the application demands platform specific features, and keeping up with Android and iOS updates. To address these conditions, developers can choose Kotlin Multiplatform (KMP), also a cross-platform solution.

As Jetbrains mentions in their official documentation, Kotlin Multiplatform is an open-source technology developed by JetBrains that enables developers to create applications for multiple platforms efficiently by reusing code across them and still having the native approach benefits[11]. Developers can share code without losing the qualities of native programming, have good user experience, good app performance and having full platform capabilities, alongside the benefits of the cross-platform, reducing development time, consistency in behavior in the different platforms[12]. In this approach, the developer has 3 options, it can share and only write a critical feature in Kotlin, or implement the full logic and only develop the UI natively, and finally, with Compose Multiplatform, also share the UI[11].

Jetpack Compose[13] is a toolkit to develop the User Interface of Android apps. It is implemented in a declarative way, compatible with Kotlin APIs, enabling developers to intuitively build fast UIs. It also supports Google's Material Design, an "open-source design system for building beautiful, usable products"[14]. Since 2021, developers can build shared UIs for Android, iOS, desktop and web, feeling natural on every platform with Compose Multiplatform.[15]

Thompson Carter, refers in his book[16] the same benefits of KMP, code re-usability, faster development, reduce maintenance (as it is only need to fix bugs in one place), flexibility (share code and still access platform-specific features and APIs) and community support. However, he mentions some challenges as:

- Maturity, as the KMP is recent and is still evolving, some libraries and tools may not be fully supported;
- The integration with Platform-Specific APIs can be challenging;
- The transition from single-platform development can be a bit difficult, with the introduction of new paradigms;
- And the integration with an existing project can involve significant re-factoring.

3.1.6 Comparative Analysis

So, on one hand, using cross-platform frameworks can be an effective solution to develop applications for multiple platforms, when time, cost and basic interoperability is the primary concern. On the other hand, these frameworks tend to be behind the evolution of Android and iOS, and cannot take full advantage of each platform's features. Robert Nagy therefore refines the development-cost equation as:

$$CostofDevelopment(n) = FC * (1 + CostofgoingNative)$$

In this equation, n is the number of platforms and FC is the feature complexity. As for the *Cost of going Native*, reflect the additional effort required when some parts must be implemented with native code. Its value depends on the level of interoperability that the cross-platform framework provides with native components, which is not optimal. When there is a necessity to implement native code, the knowledge gap between the framework and native languages adds to the expense, as well as the synchronization.[6]

With this, it is possible to visualize the different cost of development, adding the Cross-Platform with roadblocks. The Figure 3.1, illustrates how the cost of cross-platform development increases relative to both cost of Native and Native with Sync Costs. A roadblock occurs when a required feature cannot be implemented only with the frameworks and must instead be written in native code, producing a sudden rising in cost. If no native features are required, cross-platform development can remain less expensive with the previous discussed limitations. However, as more features demand native implementation, the cumulative cost eventually exceeds that of a fully native approach.[6]

The Table 3.1 syntheses the characteristics of mobile development approaches .

Figure 3.2: Cost of cross-platform development with potential roadblocks as a function complexity[6]

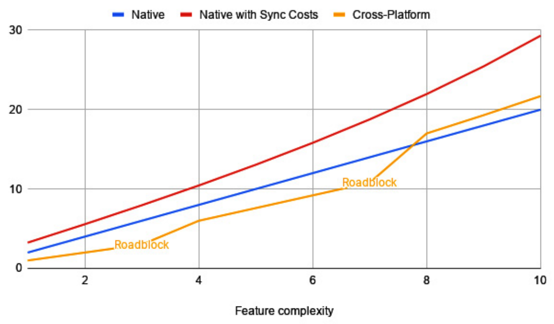


Table 3.1: Comparison of mobile application development approaches [5], [10], [12]

Approach	**Advantages**	**Disadvantages**
Native apps	Best performance; full access to platform APIs and sensors; highest UX fidelity; mature tooling and debugging.	Two codebases for multiple platforms with higher cost.
Web apps / PWAs	Single codebase; runs on desktops and mobile browsers; easy distribution via web.	Limited access to some native features; performance and energy use vary vs native.
Hybrid (wrapped web)	Faster port from an existing web app to an installable app; single web codebase; plugin ecosystems to access device features.	Worse performance than native; higher CPU and memory usage; some native features may be unavailable; UX can feel non-native.
React Native (interpreted)	High code reuse; React ecosystem; UI renders using native components; faster than pure WebView hybrids for UI responsiveness; good developer velocity.	Bridge overhead can reduce performance for heavy UI/CPU work; some features require native modules; platform parity depends on third-party library support.
Flutter (cross-compiled)	Compiled to native code (via Dart); high performance; consistent UI across platforms via Skia; rich widget set enabling polished UIs.	Larger binary sizes; occasionally needs platform channels to access native features; different paradigm (Dart and Flutter widgets) for developers.
Kotlin Multiplatform (KMP)	Share core business logic (networking, storage, domain rules) across platforms; retain native UIs or use Compose Multiplatform; near-native performance and good interoperability with native code.	Ecosystem still maturing; some libraries may be platform-limited; complex when integrating with existing native projects; tooling and third-party support are improving.

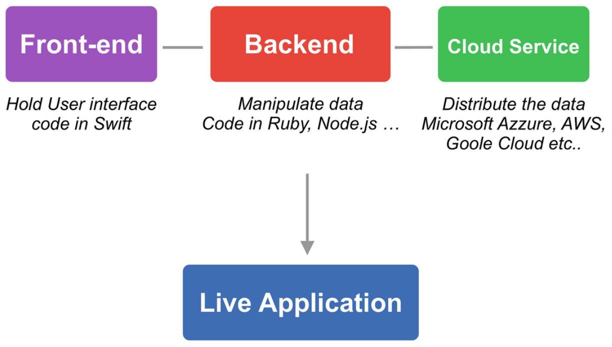
3.2 Mobile Backend Development Strategies

3.2.1 Backend Fundamentals

Mobile Applications follow a distributed architecture that has 2 components, frontend and the backend. The frontend, also called the client-side is the part of the application that runs locally in the user's device. This is the User Interface, business logic, presentation logic, that can be developed with Compose Multiplatform and Kotlin Multiplatform, for example. The backend, also called server-side, is executed outside the device, which the user does not interact with directly. This component can handle the data, reading, writing, updating, and delete data from the database, authenticate users, and more.

De Carli (2023, Chapter 2: “Introduction to Firebase”) describes a traditional mobile architecture example[17]. He uses the Figure 3.3 to represent an iOS app that has the Front-end, the user interface code in Swift (see subsection 3.1.1), the backend that can manipulate data coded in Ruby or Node.js, and a cloud service, that distributes the data in the cloud (Microsoft Azure, AWS, Google Cloud). He mentions that the application communicates with a virtual machine that can be coded in Ruby, Node.js or even PHP, that is hosted by a cloud service or, less common now, in a physical server. This is a traditional backend, which, in a complex application is usually required to have separate developers for each part of the application, including a backend developer to code the logic and APIs, and a DevOps to distribute in the cloud[17].

Figure 3.3: An iOS app built with a backend host on a cloud service[17]

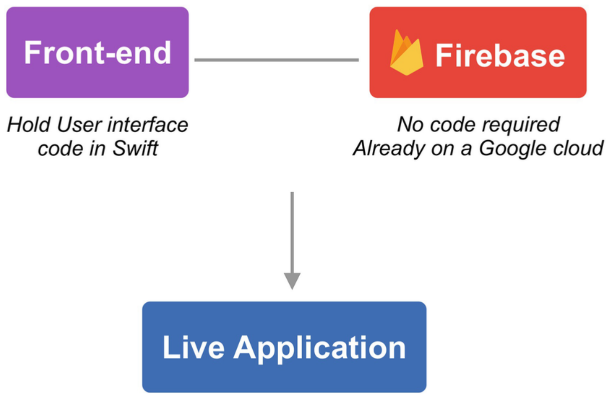

Front-end
Backend
Cloud Service
Hold User interface code in Swift
Manipulate data Code in Ruby, Node.js ...
Distribute the data Microsoft Azzure, AWS, Goole Cloud etc..
Live Application

However, a different strategy for the backend, that is more recurrent for developers that what to focus on the application development, is the use of Backend as a Service (BaaS) or Mobile Backend as a Service (MBaaS)[18]. BaaS is a cloud-based service model that provides backend features and tools needed for the mobile application that are executed in the server, such as user authentication, database, remote updating, and push notifications, as well as cloud storage and hosting[19]. Firebase[20], is an example of a BaaS, and in the second chapter of Sullivan De Caril’ book, he introduces the service and shows the Figure 3.4, that represents the iOS application integrating Firebase. With this, it is possible to see that the backend is managed by the tool and, as Firebase is from Google, it is also built in Google Cloud, having a simpler architecture. The app receives data from Firebase using the APIs provided, without needing to write code on the server side or having another provider to host[17].

BaaS platforms provide services such as user authentication, database storage, cloud functions, and real-time messaging, that speeds up development cycles and decreases the overhead involved in infrastructure management[21].

The disadvantage of these platforms include the lack of flexibility and fine-grained control, specially when the developer has complex data, custom analytics or there is proprietary infrastructure. Besides that, when choosing a platform, the app developer is subject to vendor lock-in, becoming dependent in the platforms tools, APIs and data formats. With this, developers can also go for a hybrid approach, using features form the BaaS and developing other backend

Figure 3.4: An iOS app built with Firebase's backend[17]


Front-end
Firebase
Hold User interface code in Swift
No code required Already on a Google cloud
Live Application

features with a programming language[21].

3.2.2 Data management strategies

In mobile applications data management is an important piece that influences the application performance, user experience and business logic. The ability to store, retrieve, synchronize and secure data determines the success of the majority applications. Mobile phones have their limitations depending on the model of the phone and where the user is, such as limit battery capacity, variable network connectivity, constraints in storage and other computational capabilities. A big part of applications need a stable internet connectivity to provide smooth real-time capabilities[22].

In order to guarantee that the application is always working and not rely only on server connectivity, there is the Offline-First architecture. With this approach, the application will remain usable without a reliable network connection, it will present the user with the local data instead of waiting for the first network call to complete or fail, and it will fetch data when the optimal conditions are met. So, the local data is prioritized and when connectivity is restored, the local data can be synchronized[22].

A hybrid approach for data management is also recurrent, where the application also relies on cloud storage.

Relational and non-relational databases represent two fundamentally different approaches to data management in mobile application development. Relational databases, organize data into structured tables with predefined schemas and utilize Structured Query Language (SQL) for data manipulation, providing strong ACID properties (Atomicity, Consistency, Isolation, Durability) that ensure transactional reliability and data integrity[23]. Non-relational or NoSQL databases employ flexible, schema-less data models that accommodate diverse data types without rigid table structures, offering horizontal scalability and superior performance for unstructured data and high-volume operations [23]. For mobile applications, some examples of databases include, MySQL[24], PostgreSQL[25] and SQLite[26] as Relational databases. MongoDB Realm[27], MongoDB[28], Redis[29], examples for NoSQL, as well as Firebase Realtime Database[30] and Firebase Firestore[31], with this two last being a Google Cloud solution. For local database, usual option are between Room[32], that is an abstraction layer over SQlite, SQLite, Realm or Core Data, Apple's framework for local persistence or cache on iOS[33]. The selection between these database types depends on specific mobile application requirements such as data structure complexity, offline capabilities, synchronization needs, and scalability projections, with modern applications often employing hybrid approaches that leverage both relational and NoSQL databases for different functional domains[34].

3.2.3 Data Synchronization

With data in local storage and data in an external database, data synchronization is critical in mobile applications that depend on multi-user interaction, collaboration and shared data. Synchronization ensures that all users and devices have consistent up-to-date information.

Following Zach McCormick and Douglas C.Schmidt paper in data synchronization patters[35], these patterns can be categorized into data synchronization mechanisms patterns (asynchronous vs synchronous), data storage and availability patterns (partial vs complete), and data transfer patterns (full transfer, timestamp transfer, or delta transfer).

Data synchronization mechanisms patterns address how synchronization operations execute relative to the user interface, with asynchronous synchronization performing data transfers in the background without blocking user interactions, thereby maintaining application responsiveness but requiring careful management of concurrent data modifications, while synchronous synchronization blocks the user interface until data transfer completes, guaranteeing immediate consistency at the cost of reduced responsiveness.

Data storage and availability patterns determine the scope of data maintained on mobile devices, where partial storage retains only a subset of server data based on user needs or device constraints to minimize storage requirements and reduce synchronization overhead, whereas complete storage maintains a full replica of server data on the device, enabling comprehensive offline functionality but demanding greater storage capacity and longer initial synchronization times.

Data transfer patterns define what information moves between client and server during synchronization, including full transfer which transmits entire datasets regardless of changes, timestamp transfer which exchanges only records modified since the last synchronization by comparing modification timestamps, and delta transfer which computes and transmits only the specific changes or differences between client and server versions, minimizing bandwidth consumption but increasing computational complexity.

When data is written locally while offline, there will be differences between the online data, and there will be a conflict that needs to be resolve before synchronization. There are several conflict resolution strategies, such as Server Wins or Client Wins, where it is decided if the data in the server prevails over the client or the other way, Merge conflict by implementing logic to merge changes when possible with user intervention possible needed with complex data, Vector Clocks when there is a complex scenario, helping understand the relationships between changes[36]. Another method that Androids mentions in its documentation is called Last write wins, where the device adds metadata to the written data in the network, and when the network data source receives, any older data will be discard[37]. This can result in data loss if two offline devices write conflicting updates sequentially; when both reconnect and synchronize, the data written by the first device will be discarded.

For a good user experience on unreliable networks, an offline-first hybrid architecture is a good option. Keeping a local databases for immediate responsiveness, use background workers like Android WorkManager[38] and iOS Background Tasks[39], for scheduled synchronization, and Backend as a Service platforms (Google Firebase, AWS Amplify) for cross-device consistency. Such architecture balance responsiveness, bandwidth, and storage while allowing application-level conflict-resolution policies to be enforced.

3.2.4 MBaaS Providers

There are several different solutions to choose for MBaaS, from big cloud corporations such has Google's Firebase, Amazon's AWS Amplify, Microsoft Azure, to open-source like Supabase, Parse, Back4App. Below are some relevant MBaaS:

Google Firebase

Firebase is a Backend as a Service developed by Google and built on Google Cloud infrastructure, that provides various functionalities to help build, deploy and manage mobile a and web applications. Developers do not need to build APIs, as the platform provides its dedicated Software Development Kit (SDK) to interact with its set of tools. Here are the following tools that Google's Firebase provide:

- **Cloud Firestore** - is a flexible, scalable database for mobile, web, and server development. It keeps the data in sync across client apps through realtime listeners and offers offline support for mobile and web so developers can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.[31]
- **Cloud Functions** - Firebase is a serverless framework that lets developers to automatically run backend code in response to events triggered by background events, HTTPS requests, the Admin SDK, or Cloud Scheduler jobs. JavaScript, TypeScript or Python code is stored on Google Cloud infrastructure and runs in a managed environment. There's no need to manage and scale own servers.[40]
- **Realtime Database** - is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our Apple platforms, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.[30]
- **Authentication** - provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to the app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. Integrates tightly with other Firebase services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with custom backend.[41]
- **Local Emulator Suite** - consists of individual service emulators built to accurately mimic the behavior of Firebase services. This means you can connect your app directly to these emulators to perform integration testing or QA without touching production data.[42]
- **Data Connect** - is a relational database service for mobile and web apps that lets developers build and scale using a fully-managed PostgreSQL database powered by Cloud SQL. It provides secure schema, query and mutation management using GraphQL technology that integrates well with Firebase Authentication. Developers can quickly integrate this product into the mobile and web apps with SDK support in Kotlin Android, iOS, Flutter, and web.[43]
- **Security Rules** - leverage extensible, flexible configuration languages to define what data users can access for Realtime Database, Cloud Firestore, and Cloud Storage. Firebase

Realtime Database Security Rules leverage JSON in rule definitions, while Cloud Firestore Security Rules and Firebase Security Rules for Cloud Storage leverage a unique language built to accommodate more complex rules-specific structures. Developers can write simple or complex rules that protect the app's data to the level of granularity that the developer's specific app requires.[44]

- **App Check** - helps protect the app backends from abuse by preventing unauthorized clients from accessing the backend resources. It works with both Google services (including Firebase and Google Cloud services) and custom backends to keep resources safe.[45]
- **Cloud Messaging** (FCM) is a cross-platform messaging solution that lets developers reliably send messages. It can notify a client app that new email or other data is available to sync. It can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4096 bytes to a client app.[46]
- **Other Tools** - Firebase provides a several tools for monitoring the app with Firebase Crashlytics and Firebase Performance Monitoring, to engage the user with Google Analytics, Firebase In-App Messaging, Google AdMob, Google Ads and more.[47]

To use Firebase there are two option in terms of Pricing Plans, the *Spark Plan* which is the free plan, and the *Blaze Plan* that is a pay as you go plan, meaning that, when passing a threshold in a specific feature, there is a cost associated. Here are the limits[48]:

- **Authentication** - there is no cost until 50k Monthly Active Users (MAUs), where with the Blaze Plan you pay more for each MAU where the maximum is 0.0025USD per extra MAU if there are more than 10,000,000.
- **Cloud Firestore** - the free version has 1GiB in Store data, 20k document writes per day, 50k document reads and 20k deletions. In the paid plan, when passin the limit the developer has to pay 0.039USD per 100,000 document reads, 0.117USD per 100,000 document writes, 0.013USD per 100,00 document deletions and 0.117 per GiB of storage per month, also having a backup data with 0.039USD per GiB.
- **Cloud Functions** - there is no cost until 2M invocations per month, and there are some limits for the procession power.
- **Cloud Storage** - it has 5GB in the free plan for storage, 1GB per day in download, 20k upload operations per day and 50K downloads. As for the paid plan, it cost 0.026USD per GB extra in storage, 0.12USD per GB extra downloaded and 0.05USD and 0.004USD per 10k upload and download operations respective.
- **Realtime database** - the free tier offers 1GB of storage and in the paid it is 5USD per GB extra.

Supabase

Supabase[49] is an open-source alternative to the BaaS platform Firebase, as it is a combination of different open-source tools that are scalable[50]. Supabase has the following tools:

- **Database** - a Postgres database, available for every project. The relational struture is advantageous for performing complex queries, joins and full-text searches[50].

- **Authentication** - makes it easy to implement authentication and authorization in the app. Supabase provides client SDKs and API endpoints to help create and manage users. Users can use many popular Auth methods, including password, magic link, one-time password (OTP), social login, and single sign-on (SSO).[51]
- **Storage** - open source S3 Compatible Object Store that makes it simple to upload and serve files of any size, providing a robust framework for file access controls. It can store images, videos, documents, and any other file type. Serve assets with a global CDN to reduce latency from over 285 cities globally. Supabase Storage includes a built-in image optimizer, in order to resize and compress media files on demand[52].
- **Edge Functions** - server-side TypeScript functions, distributed globally at the edge — close to the users. They can be used for listening to webhooks or integrating the Supabase project with third-parties.[53]
- **Realtime** - Supabase provides a globally distributed Realtime service that can send low-latency messages between clients, good for real-time messaging, database changes, cursor tracking, game events, and custom notifications. Track and synchronize user state across clients, ideal for showing who's online, or active participants. And listen to database changes in real-time.[54]

For features not natively provided by Supabase, such as push notifications or built-in analytics, developers can integrate third-party services, webhooks, or direct API integrations[55].

Supabase has 2 main plans, the free option and a pro option where the developer pays 25USD per month[56]. Here what a developer gets with each plan:

- **Database** - The free plan gets 500MB os database size, and the pro gets 8GB with option of paying more 0.125USD per extra GB. With the pro plan, the developer gets automatic backups of seven days, the project is never paused, has point of recovery for 100USD per month per 7 days retention, and more.
- **Authentication** - There are unlimited users, but in the free plan there is the limit of 50,000 Monthly Active Users (MAUs) and 100,000 MAUs for the pro plan, with the option of paying 0.00325USD per MAU extra. The pro plan has extra featuers.
- **Storage** - The free plan has 1GB include of storage while the paid has 100GB then 0.021USD per GB. The free plan has the max file upload size of 50MB and the paid has 500GB.
- **Realtime** - It has the current peak connection where in the free plan has 200 included and the pro plan has 500 with 10USD per 1000 more. The free has 2 Million messages per month included while the pro has 5 Million paing 2.50USD per Million extra.
- **Edge Functions** - The free plan has 500,000 invocations included while the pro plan as 2 Million with 2USD per Million extra.

Parse

Parse is an open-source Mobile Backend as a Service, that provides tools to develop the backend of mobile applications. This option was from Facebook that decided to shut down and make it open-source and now, developers have to host the service, either in platforms that can manage Parse Server or self-host in a cloud service or in private servers[57]. Parse Platform has similar features has the other services, which are[58]:

- **Database & Storage** - Supports MongoDB and PostgreSQL for structured data, with flexible file storage options including Amazon S3, Google Cloud Storage, or local filesystem storage. GridFS is commonly used as the default large-file storage when MongoDB is chosen[59]. Parse also supports multi-database deployments and can be configured to shard or separate concerns across databases for scale and isolation.
- **Real-time** - Live queries for real-time data updates and built-in push notification support across all platforms, with real time updates.
- **Authentication & Security** - Provides comprehensive user management (signup, login, password reset) with support for OAuth, email verification, and custom authentication.
- **APIs & Integration** - REST and GraphQL APIs with multi-platform SDKs for seamless integration across all major platforms. Supports custom queries and mutations.
- **Cloud Functions** - Allows server-side logic via custom Cloud Functions, database triggers that run on create/update/delete events, and background job processing for async or scheduled tasks.
- **Self-hosting Freedom** - Can be deployed anywhere that runs Node.js (cloud VMs, containers). Full control over infrastructure and data, with a Parse Dashboard for visual management and monitoring, guaranteing complete data ownership.

Parse platform and server are free and open-source, however, the cost lays in the hosting.

Back4App

Back4App is another MBaaS, cloud-hosted solution, that runs and hardened the open-source Parse Platform, exposing the same APIs and SDKs while adding management, scalability, backups and developer tools[60]. With this platform, developers can take advantage of Parse functionalities while simplifying deployment and infrastructure management. In terms of features, they are and work the same as the Parse, adding hosting, managing and UI features[61]. With Back4App there is no vendor lock-in, meaning that the developer can, anytime, export the database, files, Cloud code and run the same Parse Server elsewhere[62].

In terms of pricing, the platform has a free plan, the MVP to "Validate Ideas Quickly — Launch Fast on Our Managed Serverless Backend" (15USD per month paying annually or 25USD monthly), Pay as you go to "Run & Scale Applications on a Serverless Infrastructure" (80USD or 100USD per month) and the Dedicated to "Production-Grade Speed, Isolation & Flexibility on Dedicated Resources"(400USD or 500USD per month)[63]. Here are the differences:

- **Request per month** - in the free plan is 25k requests, while the MVP is raised to 500k with 5USD per 100k extra request, the Pay as you go 5M with 2USD per 100k extra, and the last one is unlimited.
- **Data Storage** - the free plan has 250MB then is 1GB followed by 3GB and then 8GB, with 15USD monthly for each adicional gigabyte extra.

- **Data Transfer** - the free plan allows 1GB, the MVP is 250GB, the Pay as you go is 1TB and the Dedicated is 2TB, with 0.1USD per each 1GB of adicional transfer.
- **File Storage** - the free plan has 1GB, then 50GB, 250GB and 1TB, with 1USD charge per 10GB extra.

AWS Amplify

AWS Amplify is the Amazon's cloud-powered MBaas and developer toolchain that helps mobile developers build, deploy and scale fullstack applications. It offers end-to-end workflows for developing, deploying and managing cross-platform apps, allowing seamless cloud integration.[64] Here are the features[65]:

- **Authentication** - Create seamless on-boarding flows with a fully-managed user directory and pre-built sign-up, sign-in, forgot password, and multi-factor auth workflows. Amplify also supports login with a social provider such as Facebook, Google Sign-In, or Login With Amazon and provides fine grained access control to mobile and web applications. Powered by Amazon Cognito.
- **DataStore** - Use a multi-platform on-device persistent storage engine that automatically synchronizes data between mobile/desktop/web apps and the cloud, powered by GraphQL. DataStore provides a programming model for leveraging shared and distributed data without writing additional code for offline and online scenarios, which makes working with distributed, cross-user data just as simple as working with local-only data. Powered by AWS AppSync.
- **Analytics** - Understand the behavior of the iOS or Android users. Use auto tracking to track user sessions and web page metrics or create custom user attributes and in-app metrics. Provides access to real time data stream and developers can analyze the data for customer insights and build data driven marketing strategies to drive customer adoption, engagement, and retention. Powered by Amazon Pinpoint and Amazon Kinesis.
- **API** - Make secure HTTP requests to GraphQL and REST endpoints to access, manipulate, and combine data from one or more data sources such Amazon DynamoDB, Amazon Aurora Serverless, and the apps custom data sources with AWS Lambda. Amplify enables you to easily build scalable applications that require real-time updates, local data access for offline scenarios, and data synchronization with customizable conflict resolution when devices are back online. Powered by AWS AppSync and Amazon API Gateway.
- **Functions** - Add a Lambda function to the project which the developer can use alongside a REST API or as a datasource in the GraphQL API in the Amplify CLI. The developer can update the Lambda execution role policies for the function to access other resources generated and maintained by the CLI, using the CLI. Amplify CLI enables developers to create, test and deploy Lambda functions across various runtimes.
- **PubSub** - Pass messages between the app instances and the app's backend creating real-time interactive experiences. Amplify provides connectivity with cloud-based message-oriented middleware. Powered by AWS IoT services and Generic MQTT Over WebSocket Providers.

- **Push Notifications** - Improve customer engagement by using marketing and analytics capabilities. Leverage customer insights to segment and target the customers more effectively. The developer can tailor the content and communicate through multiple channels including email, texts as well as push notifications. Powered by Amazon Pinpoint.
- **Storage** - Store and manage user generated content such as photos, videos securely on device or in the cloud. The AWS Amplify Storage module provides a simple mechanism for managing user content for the app in public, protected or private storage buckets. Leverage cloud scale storage so that is easy to take the application from prototype to production. Powered by Amazon S3.
- **Predictions and Interactions** - Enhance the app by adding AI/ML capabilities and build interactive and engaging conversational bots with the same deep learning technologies that power Amazon Alexa.

AWS Amplify provides a comprehensive toolkit for application development, with open-source libraries, powered by AWS services and can be used with Amplify CLI and Amplify Studio or an existing AWS backend. Amplify Studio brings a visual, collaborative environment for modeling data, managing users and creating backend resources. With Amplify CLI toolchain, developers can configure and maintain backends. And the ecosystem, also provides the Amplify UI Components, an open-source set of ready-to-use interface elements for authentication, storage and interactions that integrate directly with backend resources.[65]

- **Data Storage** - No cost up to 5 GB stored on CDN per month, then 0.023USD per GB per month.
- **Data Transfer Out** - Up to 15 GB per month at no cost, then 0.15USD per GB served for additional transfer.
- **Request Count (Server-Side Rendering)** - It covers up to 500,000 requests per month at no cost, then 0.30USD per 1 million requests thereafter.
- **Request Duration (SSR)** No cost up to 100 GB-hours per month, then 0.20USD per GB-hour.
- **Authentication powered by Amazon Cognito** - There are 3 plans, the Essentials, Plus and Lite that provides different features. In the Essencials, the first 10,000 monthly active users (MAUs) are for free, then is 0.0015USD per MAU. For the Plus, is cost 0.02USD per MAU. For the Lite plan, it has a free tier for the first 10,000 raising to 0.0055USD per MAU between 10,001 and 100,000. Then, until 1,000,000 is 0.0046USD per MAU, 0.00325 per MAU until 10,000,000 and greater than that is 0.0025USD.[66]
- **Real-time APIs powered by AWS AppSync** - 4.00USD per million Query and Data Modification Operations, 2.00USD per million Real-time Updates, 0.08USD per million minutes of connection to the AWS AppSync GraphQL service. There are different fees for cahing, starting at 0.044USD for 1 vCPU, 1.55GiB of memory and Low to moderate network performance to 6.775USD for 48vCPU, 317.77GiB of memory and 10Gigabit of network performance. The AppSync GraphQL Free Tier offers 250,000 query or data modification operations, 250,000 real-time updates and 600,000 connection-minutes.[67]

- **Data powered by Amazon DynamoDB** - Two pricing options available including on-demand and provisioned. DynamoDB on-demand mode is a serverless option with pay-per-request pricing and automatic scaling, with 0.705USD per million write request units (WCU) and 0.1415USD per million read request units (RCU), the first 25 GB stored per month is free using the DynamoDB Standard table class and 0.283USD per GB-month thereafter. With provisioned capacity mode, the developer must specify the number of reads and writes per second that expects the application to require, with 0.000735USD per WCU, 0.000147USD per RCU and the Data Storage is the same as the other plan.[68]
- **Functions powered by AWS Lambda** Lambda counts a request each time it starts executing in response to an event notification trigger, duration is calculated from the time the code begins executing until it returns or otherwise terminates, rounded up to the nearest 1 ms, with the price depends on the amount of memory the developer allocates to the function.[69]
- **Storage powered by Amazon Simple** - Pay only for what is used. There is no minimum charge. Amazon S3 cost components are storage pricing, request and data retrieval pricing, data transfer and transfer acceleration pricing, data management and insights feature pricing, replication pricing, and transform and query feature pricing.[70]

Microsoft Azure

Microsoft Azure used to have Azure Mobile Services, and in 2016 Microsoft announced the transition to Azure App Service[71]. Now, Azure does not provide a MBaaS like the other providers, but through multiple services in Azure App Service[72]. Azure App Services provides a fully managed platform for building, hosting and scaling web and mobile applications backends using multiple programming languages[72]. It allows integrations with the following services:

- **Azure Functions** - Execute event-driven serverless code that responds to HTTP requests, database changes or queue messages, with an end-to-end development experience for scalable APIs[73].
- **Azure Cosmos DB** - A globally distributed, NoSQL database service that provides access to data with automatic scaling and multi-region replications. It supports several APIs, such asMongoDB, PostgreSQL and Apache Cassandra[74]
- **Azure SQL Database** - A fully managed relational Database as a Service that provides a fast, flexible and elastic SQL database[75].
- **Microsoft Entra ID** - Formerly known as Azure Active Directory (Azure AD), this service manages identity and access, supporting OAuth2, OpenID Connect, and enterprise-grade authentications[76].
- **Azure Notification Hubs** - A cross-platform push notification service that allows the delivery of personalized messages to mobile devices with support for tagging and segmentation[77].
- **Azure API Management** - Enables the creation, protection, and monitoring of APIs used by mobile applications. It provides rate limiting, authentication, analytic and developer portal capabilities[78].

- **Azure Storage** - Offers scalable object, file and queue storage options for unstructured data, media files, and backend integration, forming a common storage layer for mobile applications[79].

For the pricing, each service is billed independently, and has different plans for each services with multiple price variations, including different prices for different regions.

Other Providers

Other MBaaS available include the follow[80]–[83]:

- Oracle Mobile Hub
- Backendless
- Salesforce Heroku
- Apache Usegrid
- Progress Kinvey
- Kumulos by Optimove
- AppsPanel
- PubNub
- Apple CloudKit
- BrainCloud
- Kuzzle
- Appwrite

MBaaS Provider Comparison

Table 3.2 presents a comprehensive comparison of the major MBaaS providers currently available in the market. The feature matrix demonstrates that modern MBaaS platforms have converged on a common set of core functionalities, including authentication mechanisms, real-time data synchronization capabilities, serverless computing through Cloud Functions, and comprehensive security rules. All evaluated providers offer APIs (both REST and GraphQL), file storage services, and offline support, reflecting the maturation and standardization of the MBaaS ecosystem.

However, significant differences emerge in the areas of database technology choices, open-source availability, and infrastructure control. Notably, Supabase distinguishes itself by offering only relational database technology (PostgreSQL), whereas other providers support both NoSQL and SQL databases, providing developers with flexibility in selecting the most appropriate data model for their application. Additionally, the degree of vendor lock-in varies considerably, with open-source solutions like Supabase and Parse offering greater portability and self-hosting capabilities, while proprietary platforms such as Firebase and AWS Amplify provide tightly integrated ecosystems with limited exit strategies.

Table 3.2: Feature Matrix: MBaaS Providers Comparison

Feature	**Firebase**	**Supabase**	**Parse**	**Back4App**	**AWS Amplify**	**Azure**
Auth	✓	✓	✓	✓	✓	✓
Real-time Sync	✓	✓	✓	✓	✓	✓
NoSQL DB	Firestore, Realtime DB	✗	MongoDB	MongoDB	DynamoDB	Cosmos DB
SQL DB	Data Connect	PostgreSQL	PostgreSQL	PostgreSQL	Aurora	Azure SQL
Cloud Functions	✓	✓	✓	✓	✓	✓
Push Notifications	✓	✗	✓	✓	✓	✓
APIs	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓
Offline Support	✓	✓	✓	✓	✓	✓
Analytics	✓	✗	✗	✓	✓	✗
DB Triggers	✓	✓	✓	✓	✓	✓
Security Rules	✓	✓	✓	✓	✓	✓
Open Source	✗	✓	✓	Partial	Partial	✗
Self-Hosting	✗	✓	✓	✗	✗	✗
Vendor Lock-in	High	Low	Low	Low	High	High

3.3 Related Work

To understand the existing solutions available in the Portuguese casual basketball market, it was conducted a search for applications and platforms addressing the challenges identified. This section presents the research methodology and the identified applications.

3.3.1 Methodology

By searching the browser using the query "Play basketball in Portugal", the useful links found for someone who just want to go to a public court and play some casual games, or shoot some balls, where from a website called "Courts of the World"[84], "Meetup"[85], a question in Reddit from 2021[86], a blog called "World Baller"[87], and some websites for clubs and schools. With the query "Basketball pickup games Portugal", the results were similar, with the AI from the browser telling to look for games in Facebook groups, in Meetup, and suggested 3 courts in Lisbon.

Using Perplexity AI, using this prompt in a deep research "I am from Lisbon and I play basketball. I would like to go play on the weekend in a public court. Where can I go? Is there any platform I can find some casual games to join?"[88], the model provided 4 public courts with a description, for example:

"Campo de Laranjeiras ("The Blue Court") near Laranjeiras Metro features two full courts with fresh surfaces and multiple hoops. Locals refer to it as the "blue court" due to its bright surface colour. It has a semi-competitive vibe with frequent rotation of players and is open 24/7 with free access."

Besides that, it provided information about a group in Meetup, and a facebook group. As for platforms, it indicated an app that does not available in Portugal, called "Pickup: Connect & Play

Sports”, another called “Pickup Sports (Adult League)”[89]. Then it also shows “Courts of the World”, that has an iOS application, and another app called “Fullcourt: Pickup Basketball”[90].

Using ChatGPT, with the same prompt, with the free version and the option of *Web Search* activated, it answered with 8 public courts, with the Meetup group already mention above, as well as the facebook group, and the blog “World Baller”[91]. With the *Thinking* option it provided some courts with the reference to “Courts of the World” website and the reddit question from above. It also provided the same platforms, and gave some tips[92].

Another query made in the browser to explore ways for players to play casual basketball was “*reservar campos de basquetebol portugal*” [2], where it shows some private places to book, and a platform called “AirCourts”[93], that gathers different courts from different sports to help users book them. In the website of AirCourts it was saying that they will join “Playtomic”[94], that is a platform to book courts and find players in padel and tennis.

In another search, “*Torneios amadores portugal basketball*”[3], only one result from the first page would be useful for someone who wants to play casual basketball games, the website “Jogabasket”[95].

Going to the Google’s Play Store, and searching for “Pickup” apps, several appeared, which Perplexity AI already mentioned, as well as, “Pickup: Play & Host Sports”[96], [97], “GoodRec”[98] (it is not available in Portugal), “WOOOBA”[99] and “Pick-Roll”[100].

3.3.2 Existing Projects

It was found three categories of existing platforms relevant to casual basketball in Portugal: dedicated court discovery platforms, game organization platforms, general community platforms.

Court Discovery Platforms

Courts of the World. [84]

Is a FIBA endorsed platform, available in web and in iOS, where users can find courts around the world. In the home page, the users can search for a place or click a button to view basketball courts near the user, which did not work when searching for Lisbon or Porto and even clicking in the button, it was loading for a long time. There is a section called “The World’s Favorite Basketball Courts” that shows cards with famous courts. In the menu is the option to list the courts, that has list of countries divided by continent. Going to Portugal, it appears the most popular courts. Opening the first court in the most popular, “Campo Dos Mártires Da Pátria”, there user can view on the map (which does not work), rate the court, this one having 8 ratings, an option to check in, saying the player is there, and the platform asks for the location to confirm, and it is saying that 1 player checked in last 5 years ago, and the user can save the court, with 7 players saving. Besides that, there are some photos of the court, with the last uploaded November 2024 (at the date), and any player can upload media. Players can add information about the court and see local insights, such has the weather conditions. In the

[2]book basketball courts portugal
[3]Amateur tournaments Portugal basketball

platform, there is also a blog and a section of the players, with a leaderboard of players that contribute for the platform, with courts added, photos and comments. This looks like a good option for players to find courts, and it looks like it has a feature of check in, so other players can see how full is the court. However, it looks like there is not much adherence on the part of the players.

Jogabasket. [95]

From the part of the Portuguese Basketball Federation, this website has the same purpose as the Courts of the World. In the home page, the user can find clubs, by searching a location, showing clubs as the result and their contacts. Scrolling down, the user can find some courts around Portugal, with map that works perfectly. There is also a section to find 3x3 tournaments, but, at the time of writing, there is nothing there. Finally, there are some images of the courts. This website is good for users to find courts in Portugal.

Game Organization Platforms

***Pickup Sports (Adult League)**[89], [101].* Is an app for Android and iOS where users can find leagues of more than 15 sports, including basketball. Looking in the website[89], in the "How it works section", it says to pick the sport, select a city and an area and find a games. For basketball, it is saying in the website that users can join competitive leagues, or just pick up games with friends. Opening the app, the user needs to create an account in order to search for games. However, the search was confusing, after choosing the location, it looked like it went back to the default, Los Angeles. Choosing the sport, basketball, it did not appear any game. It is possible to see a list of players, but just a few, and they are all form Los Angeles. Also, it is possible to see how many games the players went, but only one player had one game. It looks like the app is not used, specially in Portugal. In the app as a earning system, a user can create games and charge other users, earning seventy-five percent of the revenue, the rest must be going to the app.

***Fullcourt: Pickup Basketball**[90], [102].* - Is a platform to organize basketball pickup games. When creating an account, the user sets its height, wight and vertical. Then, the position, the skill level and the frequency that it plays. In the app, the user can create quick games, by inviting nearby players, selecting a court and hours. After creating, the players receive an invitation and can select if they are attending. In the game detail view, it shows who is going and the position the player plays, who is not going and who is still to accept, besides that, it has a chat. In the courts view, there is a map that shows the number of players in the area, and upcoming "Runs", which is a weekly recurring game at the same time and location. Going to Lisbon, there are in total 10 users, and one court in the map. In the profile view, it is available the win/lose record, showing the number of vitories and losses, the number of games and the percentage.

***Pickup: Play & Host Sports**[96], [97].* Is a platform to host and join games for various sports. When creating an account, the user chooses the sports and the level he plays. In the home page, the user can search games, enter a game code, and there is a dashboard, which is not showing anything, probably being games nearby, and this current location (Lisbon) it does not have games. There is also another view with a search bar, to search for games, sport and locations. However, it is not showing any game, as it is most likely to not existing in Portugal. There is also a feature that is communities, existing only sixteen and in Hong Kong, where there is a public chat and games for the members to join. Finally, it is possible to create games, choosing the sport, the level, if it is a public court or public, indoor or outdoor, individual or team game and if it is public or private. Then, select the time, the game duration, and the number of players, or teams (including the team size). After creating a game, it did not appear anywhere in the app. There is a function to create a team, that can join the games.

WOOOBA*[99].* Is another platform to create and join basketball games. When creating an account, the user can choose the position and also being a referee. The user can select the age, and the skill range, and also set who can see the profile. Anyone can create games, and they can be free or paid, and the app takes a fee when it is paid. Searching for games in Lisbon and Porto, and I could not find any.

Pick-Roll*[100].* This is and Italian application to find courts near the user and challenge the community. The app has a ranking system with prizes at the end of each season. A season is a period of 3 months when the leaderboard resets. When creating an account, the user can choose between "Player" or "Team/Minors" (when the user owns a team/minor). The features of the app include the map, to discover the courts around the user, games and events, to join games and events, or organize. Then, it has a check in feature, to let other players know that the user is in a court, and ratings, to rate the other players. Another feature is called "Fame", where players can receive fame points in the app by playing fairly. In the profile, besides the normal biography information, the player can choose a favourite shoes brand, hobby and pick theirs NBA team. Still in the profile, the user can earn medals by performing some actions in the platform, like "Update all your profile info", "Check-in first time" and others. To join a season, the user pays 2.59 euros, however the first 2 months are for free. In the home page, it is supposed to show the games, players and events, but in Lisbon, the test location, does not have any. However, in the map, it shows the courts in Lisbon, where there is one court with 3 ratings. In the court view, I can create a game. To create a game, there are 2 options, standard and ranked, where the ranked the users needs to be in the season, with the option of choosing the format of the game (how many players). Besides the game, an event can be created, where the user adds the start date and the end date. When a game is created it appears in the home page of the users nearby, and in the court view. The application has features really similar to the ones intended to do in the project, however it seems that is not used that much in Portugal.

General Community Platforms

Meetup*[85].* This platform is not only for sports, here users can find groups for any type of activity, and, inside the group, the host can create events. There is a group called "Lisbon Basketball Club"[103], that run indoor games every Monday and Sunday, paying a fee to book the court.

CHAPTER 4

Methodology

Developing a platform to help basketball players organize games and enhance their satisfaction at public courts requires a development process that consistently prioritizes user needs. To achieve this, the project adopts a User-Centred Design (UCD) ensuring that the users are involved throughout the design process. The UCD is applied across three iterations with each culminating in direct user evaluation. The iterative structure is complemented by Scrum, an Agile project management framework that promotes adaptive planning and continuous improvement, and Acceptance Test-Driven Development (ATDD), ensuring that the system's features align with user expectations and perform as intended. This chapter details the methodologies, tools and frameworks that are going to be used and followed during the design and development process, and the architecture that the platform will have.

4.1 User-Centered Design (UCD)

UCD is an iterative process focused on the users and theirs needs in each design phase, by actively involving users in the process. The goal of UCD is to address the whole user experience, and shape the application to the specific needs of target users[104]

To implement UCD systematically, there are 4 phases defined by the *ISO 9241-210:2019*[105], which establish Human-Centred Design (HCD) principals applicable to UCD. While the standards focus on HCD, UCD applies these principals with explicit focus on a defined target user population rather than broader human contexts. The four phases are:

- **Understand and specify the context of use**: This phase focuses on understanding the target users, their needs, pain points, and the environment in which they will use the system. Research methods such as user interviews, observations, and contextual analysis are employed to build a comprehensive understanding of user contexts;
- **Specify the user requirements**: With a clear understanding of users and their contexts, this phase involves gathering and documenting specific user requirements and desired features. Requirements are derived from user research findings and articulate what the

system must do to address identified user needs. Requirements are a set of guidelines that state what the functionality and capabilities of the platform system, which meet the user's needs and business's objectives. This will guide the development process, ensuring the end product is user-friendly, accessible and valuable.[106]

- **Produce design solutions to meet user requirements**: Based on validated user requirements, design solutions are created to address identified needs. This phase includes creating wire frames, prototypes, and mock-ups that translate requirements into tangible design artefacts;
- **Evaluate the designs against requirements**: Prototypes and design solutions are tested with actual users to validate that the designs meet specified requirements and user needs. Feedback from evaluation informs iterative refinements until user satisfaction and requirement fulfilment are achieved.

4.2 Scrum

Scrum is an Agile framework that helps developing complex project by organizing work into smaller, iterative cycles called Sprints[107]. The Agile methodology, which Scrum implements, divide development into phases that enables adaptive planning, rapid execution and continuous evaluation[108]. This approach emphasizes planing only necessary work and delivering small, frequent increments while gathering feedback in each cycle which will be integrated into the next phase[108]. While multiple Agile frameworks exist, Scrum is the most widely adopted due to its flexibility and applicability across projects different scales[109].

In this project, requirements gathered during the second phase of UCD (Section 4.1) are documented as User Stories. User Stories employ non-technical language to provide development context written from the user's perspective, clarifying not only what is being built, but also the why and the value it delivers. User Stories integrate naturally into Scrum, where each story is assigned to a Sprint and marked as complete when users can successfully execute the outlined task[110]. This approached focused on the user aligns with the project's emphasis on User-Centred Design throughout development.

To help implement Scrum effectively, this project uses Jira as a project management tool[111], enabling systematic planning, task assignment, and progress tracking in the development phase

4.3 Acceptance Test-Driven Development (ATDD)

While Scrum (Section 4.2) provides a framework for project management and the User Stories capture requirements, ATDD is a development methodology in which, acceptance test are written for each User Story before implementation[112]. This ensures that requirements are precisely defined and verifiable with the tests serving as executable specifications of expected behaviour. ATDD follows a test-first approach, acceptance tests are written first to define the expected behaviour, the feature is then implemented to satisfy these tests, and finally, the functionality is validated to ensure all tests pass. Acceptance Test can be written in different ways, scenario-oriented that uses a Given/When/Then template and a rule-oriented that is a check-list[113], which is the one being use in this project.

4.4 Iteration Development Process

This project follows an iterative and incremental development model structured into three iterations, each integrating UCD, Scrum and ATDD methodologies, and culminating in user evaluation to maintain User-Centred Design principals throughout development

Iteration 1: Prototype and Feature Validation

The first iteration includes all four phases of UCD (Section 4.1)

- A survey is conducted to understand the basketball players who uses public courts, including their habits, challenges and desired improvements to their experience. With that, *personas* are created representing different user needs and pain points. User scenarios are also developed for each *persona*, describing how they use public courts, what problems they encounter and when and how a platform would support them;
- Through the *personas* and theirs scenarios, requirements are gathered to respond to their needs, documenting them as User Stories and added to the Jira project Backlog;
- The prototypes of user stories are created in Figma to provide an interactive and testable design;
- Finally, users test the interactive prototype and provide feedback on the importance and relevance of the features. This feedback is use to update and prioritize the backlog for the next iterations.

In this first iteration, the test that users will perform are Task-based usability testing using interactive Figma prototype, followed by a Feature Prioritization Survey assessing feature importance and relevance;

4.4.1 Iteration 2: MVP Development and UI/UX Evaluation

In the second iteration, instead of a prototype, a Minimum Viable Product (MVP) is developed, with enough features to be usable in early stages and receive feedback. This iteration start applying both Scrum and ATDD methodologies (Sections 4.2 and 4.3) to implement the most critical user stories identified from the first iteration's feedback

For each user story, it is follow the ATDD approach, where acceptance criteria are defined, tests are written to validate these criteria, the feature is developed and then validated ensuring all tests pass. User stories are organized into sprints according to the Scrum framework, enabling iterative and continuous progress, tracking in Jira.

With a functional MVP, users evaluate the platform focusing on user experience (UX) and user interface (UI) aspects, through Task-Based Usability Testing to measure task performance, System Usability Scale (SUS) questionnaire combined with think-aloud protocol during task execution, and User Experience questionnair, providing feedback to guide refinements in the final iteration.

4.4.2 Iteration 3: Complete Application and Overall Evaluation

Following feedback from the second iteration, in the final one it is developed the complete platform, continuing to apply Scrum and ATDD. Improvements from previous feedback are made and all remaining user stories are implemented, developing as well the tests first, ensuring consistent quality. Users conduct a comprehensive evaluation of the complete platform, assessing overall functionality, usability, reliability and satisfaction with th final product, using SUS, task-based testing, and semi-structured interviews.

4.5 Architecture

In Android Developers "Guide to app architecture"[114], it is recommended an architecture, that includes at least two layers, the UI Layer and the Data Layer, and optionally the Domain Layer.

The UI Layer, or presentation layer, as the name implies, is the user interface and will display the application data on the screen, which should be updated whenever the data changes, either through user interaction or external events. This layer includes UI elements that render the data on the screen and state holders in order to keep and expose data to the UI, handling logic.

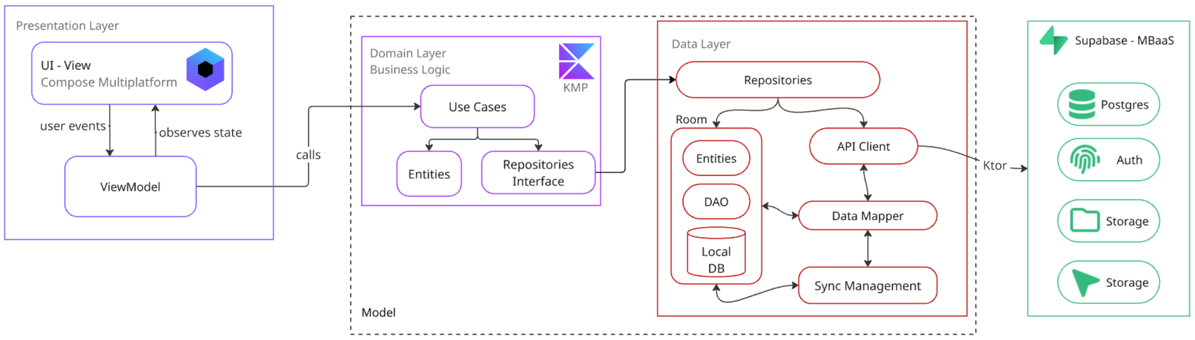
The Data layer, contains business logic that determines how the application creates, stores and changes data. This layer has repositories, that are classes responsible to expose data to the rest of the application, centralizing changes to the data resolving conflicts between data sources and abstracting the data sources. These data sources, also included in the data layer, are responsible of working with only one source of data, this being a file, database, network source.

The optional Domain Layer, sits between the UI and Data layers, encapsulating complex business logic or simpler logic which is reused by multiple view models. Classes on this layer are usually called "use cases" or "interactors" being responsible for a single functionality or user story.

For this project it was followed this guide and, besides that, it will use the Model-View-Model (MVVM) pattern, where there is a clear separation between the logic and the UI, enabling a better development, testing and a clean code base. The View is responsible for the UI and the presentation to the user, it informs the ViewModel of user actions, and observes the ViewModel states, not contain any application logic, while the ViewModel exposes the data that the View is observing, serving a link between the Model and the View. Finally, the Model abstracts the data sources, working together with ViewModel to get and save data, has the business logic of the application. This can be represented in the layers, with the UI components being the View, and the Domain and Data layer being the Model.

The Figure 4.1 represents the project's architecture and some technologies that will be used, decided after the gathering of requirements. In the Presentation Layer there is the UI components will be developed using Compose Multiplatform, and the ViewModel, classes that hold states of the application and exposes them to the View. The ViewModel will call the Use

Figure 4.1: Architecture Diagram of the Multiplatform Application



Cases, in the Domain Layer which use entities and the repositories interfaces to connect with the data layer. Here will be use Kotlin Multiplatform taking advantage of the native capabilities while working in both Android and iOS.

In the Data layer, there are the repositories, as the guide mentions[114] which will work with a local and an external data source. The local data source will be Room[32] that is part of Android Jetpack suite, making a good option for this project, being an abstraction of SQLite database, that integrates naturally with flows and coroutines, providing a annotation-based API. Compare to SQLDelight, a considerable option, it has slight more boilerplate with entities and DAOs, and less control over raw SQL, making debugging complex queries harder.

For the external data source it will be use the Supabase, an open source MBaaS with several services available that will help with the project, including PostgreSQL Database that can handle complex relationship connections while offering built-in realtime features for realtime updates. Besides that, it has authentication, storage, and server-side functions, however, lacks native push notifications and mobile analytics, which forces third-party tools integrations. Supabase can be self-hosted or use theirs services that has different pricing options, with the free tier being enough for this project, as it has a limit of 50,000 monthly active users. Comparing to others, it is not a "black box" being easy to debug and has less vendor lock-in, with a PostgresSQL database it can be run anywhere and can be exported.

Using Ktor to communicate with Supabase being a standard and recommend HTTP client, with multiple examples from Google, Jetbrains and other communities using Ktor as example in Kotlin Multiplatform projects. Another good option would be the Supabase Kotlin Client, however, this would make the application dependable on Supabase and if later the MBaaS was to be replaced, it would be more complex to migrate. With this, a little more code needs to be developed with Ktor, than Supabase Kotlin Client, but ideal to not get vendor lock-in. As some feature of the platform will need to work while the user is off-line or with bad connection, the application will have a synchronization manager that will check what needs to be updated in the local database and resolve conflicts. Finally a data mapper to convert data from Supabase to the Room database is needed.

References

[1] *Fiba world ranking for men presented by nike- fiba basketball.* [Online]. Available: https://www.fiba.basketball/en/ranking/men.

[2] *Ticha penicheiro no women's basketball hall of fame.* [Online]. Available: https://www.fpb.pt/noticia/ticha-penicheiro-no-womens-basketball-hall-of-fame-2/.

[3] *«é um bom momento para o basquetebol em portugal, queremos mais espaço» | maisfutebol.* [Online]. Available: https://maisfutebol.iol.pt/basquetebol/selecao/e-um-bom-momento-para-o-basquetebol-em-portugal-queremos-mais-espaco.

[4] *Mobile operating system market share portugal | statcounter global stats.* [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/portugal#mobile_os_combined-PT-#yearly-2009-2025-bar.

[5] M. K. Khachouch, A. Korchi, Y. Lakhrissi, and A. Moumen, «Framework choice criteria for mobile application development», *2nd International Conference on Electrical, Communication and Computer Engineering, ICECCE 2020*, Jun. 2020. DOI: 10.1109/ICECCE49384.2020.9179434.

[6] R. Nagy, *Simplifying Application Development with Kotlin Multiplatform Mobile : Write Robust Native Applications for IOS and Android Efficiently.* 2022, p. 184, ISBN: 9781801819657.

[7] S. Huber, L. Demetz, and M. Felderer, «Pwa vs the others: A comparative study on the ui energy-efficiency of progressive web apps», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12706 LNCS, pp. 464–479, 2021, ISSN: 16113349. DOI: 10.1007/978-3-030-74296-6_35. [Online]. Available: https://www.scopus.com/pages/publications/85111163302?origin=resultslist.

[8] M. Gerges and A. Elgalb, «Comprehensive comparative analysis of mobile apps development approaches», *Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023*, vol. 6, pp. 430–437, 1 Dec. 2024, ISSN: 3006-4023. DOI: 10.60087/JAIGS.V6I1.269. [Online]. Available: https://newjaigs.org/index.php/JAIGS/article/view/269.

[9] W. S. El-Kassas, B. A. Abdullah, A. H. Yousef, and A. M. Wahba, «Taxonomy of cross-platform mobile applications development approaches», *Ain Shams Engineering Journal*, vol. 8, no. 2, pp. 163–190, 2017, ISSN: 2090-4479. DOI: https://doi.org/10.1016/j.asej.2015.08.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447915001276.

[10] D. Zou and M. Y. Darus, «A comparative analysis of cross-platform mobile development frameworks», in *2024 IEEE 6th Symposium on Computers & Informatics (ISCI)*, 2024, pp. 84–90. DOI: 10.1109/ISCI62787.2024.10667693.

[11] JetBrains, *Kotlin multiplatform – build cross-platform apps.* [Online]. Available: https://www.jetbrains.com/kotlin-multiplatform/.

[12] M. Janakaraj, «Architecting shared logic with kotlin multiplatform mobile: Benefits, challenges, and ecosystem positioning», *European Journal of Computer Science and Information Technology*, vol. 13, pp. 121–141, 44 May 2025, ISSN: 20540957. DOI: 10.37745/EJCSIT.2013/VOL13N44121141.

[13] Google, *Jetpack compose ui app development toolkit - android developers.* [Online]. Available: https://developer.android.com/compose.

[14] Google, *Material design 3 - google's latest open source design system.* [Online]. Available: https://m3.material.io/.

[15] JetBrains, *Compose multiplatform – beautiful uis everywhere.* [Online]. Available: `https://www.jetbrains.com/compose-multiplatform/`.

[16] T. Carter, *Kotlin Multiplatform Projects: Sharing Code Between Android, iOS, and the Web.* Lincoln Publishers, Jul. 2025, Kotlin in Depth: Coroutines, Compose, and Multiplatform Apps for Android and Beyond.

[17] S. D. Carli, «Introduction to firebase», in *Build Mobile Apps with SwiftUI and Firebase.* Apress, 2023, pp. 17–42. DOI: `10.1007/978-1-4842-9452-9_2`. [Online]. Available: `https://learning.oreilly.com/library/view/build-mobile-apps/9781484294529/html/533376_1_En_2_Chapter.xhtml`.

[18] J.-H. Kuo, H.-M. Ruan, C.-Y. Chan, and C.-L. Lei, «Investigation of mobile app behaviors, from the aspect of real world mobile backend system», Oct. 2017, pp. 1–6. DOI: `10.1109/AEECT.2017.8257762`.

[19] P. R. Saraf, «A review on firebase (backend as a service) for mobile application development», *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, pp. 967–971, 1 Jan. 2022. DOI: `10.22214/IJRASET.2022.39958`.

[20] *Firebase.* [Online]. Available: `https://firebase.google.com/`.

[21] R. Shah, S. Jagtap, and V. Jain, «Architecting analytics-driven mobile ecosystems: Scalable backend frameworks for intelligent data flow and real-time user insights», *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 6, no. 2, pp. 83–91, Apr. 2025. DOI: `10.63282/3050-9262.IJAIDSML-V6I2P109`. [Online]. Available: `https://ijaidsml.org/index.php/ijaidsml/article/view/148`.

[22] V. R. Guda, «Implementation of offline-first architectures for android internet-based chat systems», ISSN: 2582-7138. DOI: `10 . 54660 / .IJMRGE . 2025 . 6 . 3 . 1200 - 1203`. [Online]. Available: `www . allmultidisciplinaryjournal.com`.

[23] M.-T. Chiriac, «The advantage of nosql databases over sql databases», *Database Systems Journal*, vol. XVI, p. 2025,

[24] MySQL, *Mysql*, MySQL Documentation, Oct. 2025. [Online]. Available: `https://www.mysql.com/`.

[25] PostgreSQL, *Postgresql*, PostgreSQL Documentation, Oct. 2025. [Online]. Available: `https : / / www . postgresql.org/`.

[26] SQLite, *Sqlite*, SQLite Documentation, Oct. 2025. [Online]. Available: `https://www.sqlite.org/`.

[27] M. Realm, *Mongodb realm*, MongoDB Realm Documentation, Oct. 2025. [Online]. Available: `https : //realm.netlify.app/`.

[28] MongoDB, *Mongodb*, MongoDB Documentation, Oct. 2025. [Online]. Available: `https://www.mongodb.com/`.

[29] Redis, *Redis*, Redis Documentation, Oct. 2025. [Online]. Available: `https://redis.io/`.

[30] Google Firebase, *Firebase realtime database*, Firebase Documentation, Oct. 2025. [Online]. Available: `https://firebase.google.com/docs/database`.

[31] Google Firebase, *Cloud firestore*, Firebase Documentation, Oct. 2025. [Online]. Available: `https : // firebase.google.com/docs/firestore`.

[32] Android Developers, *Save data in a local database using room*, Android Jetpack Documentation, Oct. 2025. [Online]. Available: `https://developer.android.com/training/data-storage/room`.

[33] Apple Developer, *Core data*, Core Data Documentation, Oct. 2025. [Online]. Available: `https : // developer.apple.com/documentation/coredata/`.

[34] T. Taipalus, «Database management system performance comparisons: A systematic literature review», *Journal of Systems and Software*, vol. 208, p. 111 872, Feb. 2024, ISSN: 0164-1212. DOI: `10 . 1016 / J . JSS . 2023 . 111872`. [Online]. Available: `https : / / www . sciencedirect . com / science / article / pii / S0164121223002674`.

[35] Z. Mccormick and D. C. Schmidt, «Data synchronization patterns in mobile application design, proceedings of the pattern languages of programs (plop) 2012 conference»,

[36] S. Vishvnath, *Designing a robust data synchronization system for multi-device mobile applications*, Aug. 2024. [Online]. Available: https://medium.com/@engineervishvnath/designing-a-robust-data-synchronization-system-for-multi-device-mobile-applications-c0b23e4fc0cb.

[37] *Conflict resolution*. [Online]. Available: https://developer.android.com/topic/architecture/data-layer/offline-first#conflict-resolution.

[38] Android, *Task scheduling*, Work Manager, Oct. 2025. [Online]. Available: https://developer.android.com/develop/background-work/background-tasks/persistent.

[39] A. Developer, *Background tasks*, Background Tasks docs, Oct. 2025. [Online]. Available: https://developer.apple.com/documentation/backgroundtasks.

[40] Google Firebase, *Cloud functions*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/functions.

[41] Google Firebase, *Authentication*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/auth.

[42] Google Firebase, *Local emulator suite*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/emulator-suite.

[43] Google Firebase, *Data connect*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/data-connect.

[44] Google Firebase, *Security rules*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/rules.

[45] Google Firebase, *App check*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/app-check.

[46] Google Firebase, *Cloud messaging*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/cloud-messaging.

[47] Google Firebase, *Learn about products to help you run your app*, Firebase Documentation, Oct. 2025. [Online]. Available: https://firebase.google.com/docs/run.

[48] Google Firebase, *Pricing plans*, Firebase Pricing plans, Oct. 2025. [Online]. Available: https://firebase.google.com/pricing.

[49] *Supabase*. [Online]. Available: https://supabase.com/.

[50] A. Z. Ayezabu, «Ayezabu amanuel supabase vs firebase: Evaluation of performance and development of progressive web apps», 2022.

[51] Supabase, *Auth*, Supabase Documentation, Oct. 2025. [Online]. Available: https://supabase.com/docs/guides/auth.

[52] Supabase, *Storage*, Supabase Documentation, Oct. 2025. [Online]. Available: https://supabase.com/docs/guides/storage.

[53] Supabase, *Edge funtions*, Supabase Documentation, Oct. 2025. [Online]. Available: https://supabase.com/docs/guides/functions.

[54] Supabase, *Realtime*, Supabase Documentation, Oct. 2025. [Online]. Available: https://supabase.com/docs/guides/realtime.

[55] Supabase, *Integrations*, Supabase Pricing, Oct. 2025. [Online]. Available: https://supabase.com/partners/integrations.

[56] Supabase, *Pricing*, Supabase Pricing, Oct. 2025. [Online]. Available: https://supabase.com/pricing.

[57] *What is parse? | back4app*. [Online]. Available: https://www.back4app.com/parse.

[58] Parse, *Parse features*, Parse Platform, Oct. 2025. [Online]. Available: https://parseplatform.org/#features.

[59] *Gridfs*. [Online]. Available: https://www.mongodb.com/docs/manual/core/gridfs/.

[60] *Parse on back4app*. [Online]. Available: https://www.back4app.com/product/parse-baas.

[61] *Back4app.* [Online]. Available: https://www.back4app.com/.

[62] *2025's best firebase alternatives: Open-source & scalable solutions.* [Online]. Available: https://blog.back4app.com/firebase-alternatives/.

[63] *Back4app price.* [Online]. Available: https://www.back4app.com/pricing/backend-as-a-service.

[64] *Aws amplify.* [Online]. Available: https://docs.amplify.aws/android/.

[65] *Aws amplify features.* [Online]. Available: https://aws.amazon.com/amplify/features/.

[66] *Amazon cognito pricing.* [Online]. Available: https://aws.amazon.com/cognito/pricing/.

[67] *Aws appsync pricing.* [Online]. Available: https://aws.amazon.com/appsync/pricing/.

[68] *Amazon dynamodb.* [Online]. Available: https://aws.amazon.com/dynamodb/.

[69] *Aws lambda pricing.* [Online]. Available: https://aws.amazon.com/lambda/pricing/.

[70] *Amazon s3.* [Online]. Available: https://aws.amazon.com/s3/pricing/.

[71] *Announcing azure mobile services transition to azure app service.* [Online]. Available: https://azure.microsoft.com/en-us/blog/transition-of-azure-mobile-services/.

[72] *Microsoft azure app service.* [Online]. Available: https://azure.microsoft.com/en-us/products/app-service.

[73] *Azure functions.* [Online]. Available: https://azure.microsoft.com/en-us/products/functions.

[74] *Azure cosmos db.* [Online]. Available: https://learn.microsoft.com/en-us/azure/cosmos-db/introduction.

[75] *Azure sql database.* [Online]. Available: https://azure.microsoft.com/en-us/products/azure-sql/database.

[76] *Microsoft entra id.* [Online]. Available: https://learn.microsoft.com/en-us/entra/identity/.

[77] *Azure notification hubs.* [Online]. Available: https://learn.microsoft.com/en-us/azure/notification-hubs/notification-hubs-push-notification-overview.

[78] *Azure api management.* [Online]. Available: https://learn.microsoft.com/en-us/azure/api-management/api-management-key-concepts.

[79] *Azure storage.* [Online]. Available: https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction.

[80] *Top 10 backend platforms for mobile apps.* [Online]. Available: https://blog.back4app.com/backend-platform-for-mobile-apps/.

[81] *Best mobile backend-as-a-service (mbaas) software.* [Online]. Available: https://www.g2.com/categories/mobile-backend-as-a-service-mbaas.

[82] *Mobile backend as a service (mbaas) list (november 2025).* [Online]. Available: https://www.saasworthy.com/list/mobile-backend-as-a-service-mbaas.

[83] *21 best mbaas software picks of 2025.* [Online]. Available: https://thectoclub.com/tools/best-mbaas-software/.

[84] *Basketball courts of the world – courts of the world.* [Online]. Available: https://www.courtsoftheworld.com/.

[85] *Find events & groups in portugal.* [Online]. Available: https://www.meetup.com/find/portugal/.

[86] *Anywhere to play pickup basketball in lisbon?* [Online]. Available: https://www.reddit.com/r/lisboa/comments/qtokhf/anywhere_to_play_pickup_basketball_in_lisbon/.

[87] *Pickup basketball in every city - lisbon.* [Online]. Available: https://www.worldballer.com/europe/lisbon.

[88] *I am from lisbon and i play basketball. i would like to go play on the weekend in a public court. where can i go? is there any platform i can find some casual games to join?* [Online]. Available: https://www.perplexity.ai/search/i-am-from-lisbon-and-i-play-ba-8VmBq1GuQFGTna4LnR3VXg#0.

[89] *Pickup sports.* [Online]. Available: https://pickup-sports.com/.

[90] *Fullcourt: Pickup basketball.* [Online]. Available: https://play.google.com/store/apps/details?id=com.fullcourt.fullcourt%5C&hl=en.

[91] *Lisbon basketball courts.* [Online]. Available: https://chatgpt.com/s/t_68ef91bd97b881918a2f64248073502a.

[92] *Lisbon basketball courts games.* [Online]. Available: https://chatgpt.com/s/t_68ef915fde548191a794cd0d4dbdbd67.

[93] *Aircourts - reserva grátis 1,500 campos de padel, ténis e futebol em portugal.* [Online]. Available: https://www.aircourts.com/.

[94] *Join the community and book courts online in one app | playtomic.* [Online]. Available: https://playtomic.com/.

[95] *Lisbon basketball courts games.* [Online]. Available: http://www.jogabasket.pt/.

[96] *Pickup: Play & host sports.* [Online]. Available: https://www.pickupgames.app/.

[97] *Pickup: Play & host sports play store.* [Online]. Available: https://play.google.com/store/apps/details?id=com.testpickup.app.

[98] *Goodrec.* [Online]. Available: https://play.google.com/store/apps/details?id=com.lewisblack.JustPlay.

[99] *Wooba.* [Online]. Available: https://play.google.com/store/apps/details?id=io.woooba.

[100] *Pick-roll.* [Online]. Available: https://play.google.com/store/apps/details?id=com.iperskill.pickroll.

[101] *Pickup sports play store.* [Online]. Available: https://play.google.com/store/apps/details?id=com.pickup.sports.

[102] *Fullcourt: Pickup basketball.* [Online]. Available: https://www.50four.com/.

[103] *Lisbon basketball club | meetup.* [Online]. Available: https://www.meetup.com/lisbon-basketball-meetup/.

[104] Interaction Design Foundation - IxDF, *What is user centered design (ucd)?*, Accessed: Nov.17, 2025, 2025. [Online]. Available: https://www.interaction-design.org/literature/topics/user-centered-design.

[105] *Iso 9241-210:2019 - ergonomics of human-system interaction — part 210: Human-centred design for interactive systems.* [Online]. Available: https://www.iso.org/standard/77520.html.

[106] Interaction Design Foundation - IxDF, *What are functional requirements?*, Accessed: Nov.19, 2025, 2025. [Online]. Available: https://www.interaction-design.org/literature/topics/functional-requirements.

[107] *Scrum | atlassian.* [Online]. Available: https://www.atlassian.com/agile/scrum.

[108] *What is agile?* [Online]. Available: https://www.atlassian.com/agile.

[109] *5 alternative agile methodologies other than scrum | do async | april 4, 2024.* [Online]. Available: https://doasync.com/blog/5-alternative-agile-methodologies-other-than-scrum/.

[110] *User stories with examples and a template | atlassian.* [Online]. Available: https://www.atlassian.com/agile/project-management/user-stories.

[111] *Jira | issue & project tracking software | atlassian.* [Online]. Available: https://www.atlassian.com/software/jira.

[112] A. Alliance, *Acceptance test driven development (atdd)*, 2022. [Online]. Available: https://agilealliance.org/glossary/atdd/.

[113] *Acceptance criteria for user stories in agile: Purposes, formats, examples, and best practices*. [Online]. Available: https://www.altexsoft.com/blog/acceptance-criteria-purposes-formats-and-best-practices/.

[114] A. Developers, *Guide to app architecture | app architecture | android developers*. [Online]. Available: https://developer.android.com/topic/architecture.

APPENDIX A

Additional content