# Project Scope Statement

# “ Raiddon”

ISIKA – Formation aux métiers du numérique – AL17
Axyus – Transformation Digitale

**By:**
**Farah Gauduin**
**farah.gauduin@gmail.com**

TABLE OF CONTENTS

**LIST OF FIGURES**

# Acknowledgments

*« En premier lieu, je tiens à remercier Patrick Rakotomalala et Natacha Rakocevic pour la confiance qu'ils m'ont accordé en acceptant mon intégration à Isika et à sa formation de haute qualité. Leur grande disponibilité et soutien ont bien facilité ces 10 derniers mois. Grace à Isika j'ai découvert une nouvelle passion dans la vie.*

*Je voudrais également remercier tous les encadrants tout au long de cette formation. Notamment Vincent, Pierre, Alix , Patrice et Camille.*

*Un grand merci à Axyus pour avoir accepté de financer la formation et de m'embaucher pour un contrat professionnel.*

*Un remerciement spécial à mon mari, Hermann. C'est grâce à lui que j'ai trouvé le courage, la motivation et la patience d'aboutir ce travail jusqu'à la fin.*

*Un dernier merci à mes collègues qui sont devenus des amis pour avoir partagé ces 10 mois de formation intensive avec moi. »*

# ABSTRACT

World of Warcraft addons can make everyday activities like questing and trading much easier. Blizzard's critically acclaimed MMORPG has a habit of hiding a lot of useful data it collects, and picking up a mod that offers a peek behind the curtain can be a literal gamechanger when going up against deadly bosses. Of course, the best WoW addons aren't just aimed at avid raiders. Some are more focused on utility, freshening up the multiplayer game's geriatric UI, while others help track material nodes for those all-important professions.
Raiddon is a world of warcraft addon/website aiming to provide helpful features for world of warcraft classic players. It mainly offers services that many players may find useful and that do not exist within the default game interface.

Les addons de World of Warcraft peuvent rendre les activités quotidiennes comme les quêtes et le commerce beaucoup plus faciles. Le MMORPG a l'habitude de cacher beaucoup de données utiles qu'il collecte. Bien sûr, les meilleurs addons WoW ne sont pas seulement destinés aux raideurs passionnées. Certains sont plus axés sur l'utilité, rafraîchissant l'interface utilisateur gériatrique du jeu multijoueur, tandis que d'autres aident à suivre les nœuds matériels pour ces professions très importantes.
Raiddon est un addon / site Web de World of Warcraft visant à fournir des fonctionnalités utiles aux joueurs classiques de World of Warcraft. Il propose principalement des services que de nombreux joueurs peuvent trouver utiles et qui n'existent pas dans l'interface de jeu par défaut.

# Chapter 1  Objectives and structure of the project

## 1. The Foundations of the project

Raiddon is a project that fosters all the developpement and conceptual understandings acquired throughout the ISIKA 10-month educational program. The key points that the project should satisfy are detailed in the following sections.

### 1.1 Context and architecture of the project

Figure 1 show the architecture that the project must follow. Raiddon requires the development of a modular application that amounts to drawing a parallel with an assembly of different blocks, each fulfilling very specific functions and having as few outward dependencies as possible. This allows to have several variants of the same application and therefore to be able to present the same application with different functionalities. In other words, the project is broken down into multiple micro-services, each having its own logic, that are as autonomous as possible; therefore, guaranteeing Raiddon better maintainability and scalability.

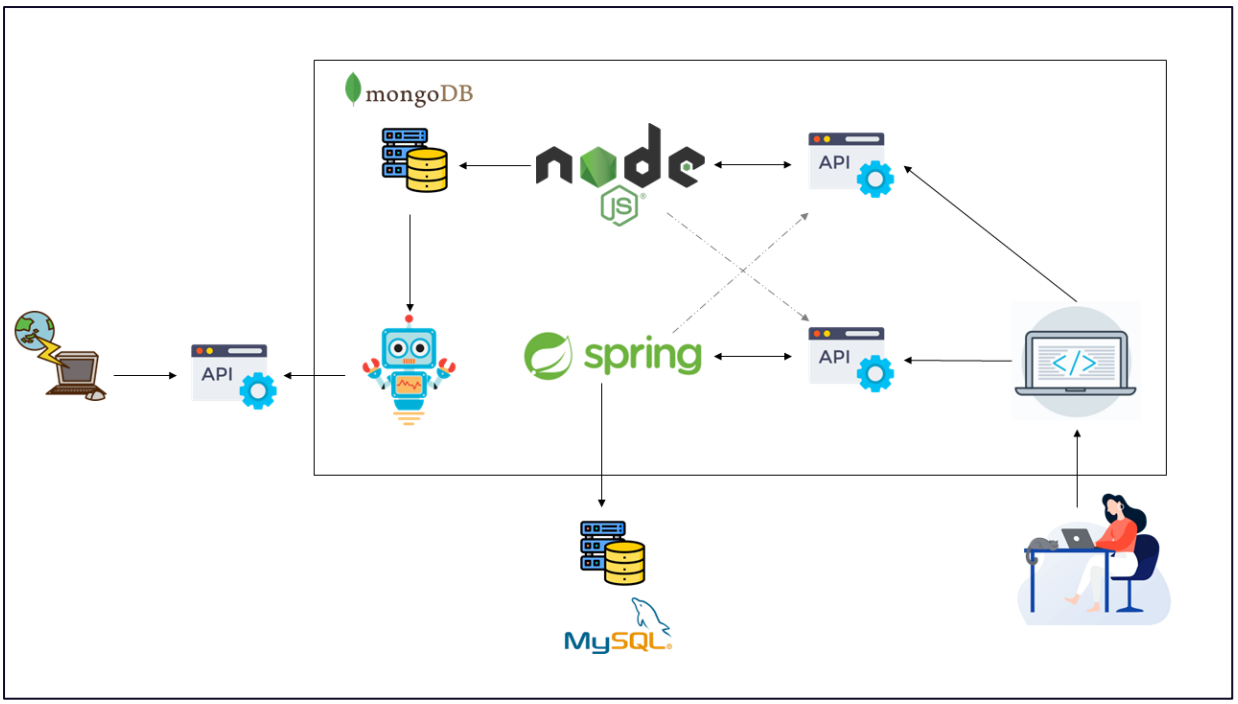



*Figure 1. Architecture of the microservices of the project.*

### 1.2 Front-End

The scripting language JavaScript is a core technology in web development, along with HTML and CSS. And its surge to the top of the ranks of the world's most prominent programming languages is tied to the rise of the World Wide Web itself. With the popularity of JavaScript comes the increase in the number of front-end frameworks, the most common of which are Angular, Ember.js, Preact, React, Svelte, and Vue.js. Based on results from Stack Overflow's survey, the most broadly used of these frameworks include React, Angular, and Vue.js.

Since Angular framework has been tackled during the Isika courses, Raiddon will be developed user the latter framework.

### 1.3 Back-End

Raiddon will include multiple microservices:

- Two NodeJS microservices able to exploit data present in a MongoDB database.
- A Spring boot application able to exploit data present in a SQL database.

## 2. Objectives of the project

World of Warcraft is a massively multiplayer online role-playing game released in 2004 by Blizzard Entertainment. In world of warcraft, a raid is an organized event in order to down bosses of a specific dungeon. Raids are formed of up to 40 players, so they are pretty hard to put in place. The main objective of a raid is to show off a guild progress of the ongoing game phase and also to gather items dropped from each boss in order to strengthen the players statistics.

Raiddon is a website aiming to provide helpful features for world of warcraft classic players. It mainly provides an intuitive and technical solution that connects users who are looking to recruit for or join a guild or a raid. The main objective is to offer services that many players may find useful and that do not exist within the default game interface.

## 3. Raiddon users

All services offered by Raiddon vary based on the status of the player using the website. A user may be registered as:

- A guild master or officer recruiting new members in his guild or looking for players to complete his raid roster on a given date and time.
- A guild member looking for a raid or a new guild.
- A player without a guild looking for a guild or a raid to join.

## 4. Raiddon services

After adding at least one playable character, a Raiddon user may manage their account in the following ways:

- Define availability to join a guild,
- Define availability to participate to a raid,
- Delete Account,
- Update Account,
- Manage equipment,
- Manage professions,
- Manage reputations,
- Add new playable character,
- Manage playable characters,
- Search guild list,
- Contact guild masters,
- Apply to join a guild,
- Search available raid list,
- Apply to join a raid,
- Search players list,

- Check attended raids history,
- Create a Guild,
- Organize a Raid Manage Guild,
- Delete player,
- Upgrade player's rank,
- Downgrade player's rank,
- Check players available for raid list,
- Check players looking for guild list,
- Recruit new player,
- Check guild members list,
- Check guild raid history.

## 5. User issues

- **Guild masters:**
  Running a guild is an undertaking and a personal investment, especially when you're just starting out. A guild master has to be active and around as often as possible for their guild to thrive. Being a guild master requires also organising raids at least once a week depending on their guild type. Many options can be found in-game (i.e., the guild interface) , however they remain very limited due to their restriction to connecting to the internet.
- **Players without guild:**
  Not having a guild in world of warcraft is indeed challenging when it comes to find a raid. Places in raids are often very hard to get and sometimes impossible.

## 6. Raiddon logo


***Figure 2. Raiddon Logos.***

# Chapter 2 - Naming Conventions

7. **Naming conventions:**
   a. **World of Warcraft (WoW):** World of Warcraft (WoW) is a massively multiplayer online role-playing game (MMORPG) released in 2004 by Blizzard Entertainment.

   b. **Guild:** In video games, a clan, community, guild or faction is an organized group of video game players that regularly play together in one or more multiplayer games.

   c. **Guild master or leader:** A guild leader is a player character who is the head of a guild. They have administrative control of the guild's operations (via "Guild Control" button on the Guild list window), including giving ranks, privileges, adding/removing guild members, etc.

   d. **Guild officer:** An Officer is a rank of member within a Guild that usually has similar privileges to the guild leader. There can be multiple ranks of officers in a guild with varying levels of privilege controlled solely by the guild's leader.

   e. **Guild Tabard:** A tabard refers to a type of clothing bearing a symbol or design and worn in the tabard equipment slot over a character's chest armor. Originally a purely cosmetic item intended to add individuality to a character's look, similar to a shirt, they now often signal an adventurer is championing a cause and thus offer reputation for wearing one in a dungeon.

   f. **Playable characters:** Players control a character avatar within a game world in third- or first-person view, exploring the landscape, fighting various monsters, completing quests, and interacting with non-player characters (NPCs) or other players.

   g. **Faction:** To create a new character, players must choose between the opposing factions of the Alliance or the Horde. Characters from the opposing factions can perform rudimentary communication (most often just "emotes"), but only members of the same faction can speak, mail, group and join guilds. The player selects the new character's race, such as orcs or trolls for the Horde, or humans or dwarves for the Alliance. Players must select the class for the character, with choices such as mages, warriors, and priests available. Most classes are limited to particular races.

   h. **Race:** Race is a term used in World of Warcraft to split various unique creatures into separate groups. Numerous races populate Azeroth including elves, trolls, human, orcs, gnomes, and murlocs, among many others. Many races can interbreed including humans, high elves, night elves, ogres, orcs, and draenei — producing offspring often called half-breeds.

   i. **Class:** A class is the primary adventuring style of a player character. A character's class determines the abilities, powers, skills, and spells they will gain throughout their adventures, and consequently the styles of play available to the character. It determines the types of weapons and armor they can use, which attributes they will value (and how those attributes function), as well as what combat roles the character is suitable for. Class also reflects a

significant choice of path for a character: whether they have chosen to pursue the dark arts of the warlock or the Holy Light of the paladin; the bloody honor of the warrior or the arcane knowledge of the mage.

j. **Profession:** A profession is a trade-oriented set of skills that player characters may learn and incrementally advance in order to gather, make, or enhance items that can be used in World of Warcraft gameplay. In essence, professions are 'jobs' characters may have. Professions are learned and improved via a trainer for a nominal fee, or sometimes advanced with special recipes. Any profession can be learned regardless of a character's faction, race, or class, although some racial traits provide bonuses to a particular profession.

k. **Reputation:** You can gain or lose favour, otherwise known as reputation, with many of the several different factions in Azeroth. Higher reputation gives access to special rewards or new quests to accomplish.

l. **Items:** An item is something that a World of Warcraft player character can carry, either in their inventory, represented by an inventory icon, or tracked on a page in the character sheet.

m. **Raid:** Raids groups are a way to have parties of more than 5 and up to 40 people, divided into up to 8 groups of up to 5 players. The terms "raid" and "raiding" primarily and traditionally refer to PvE raid-specific instances. Raid instances require playing as a team, and are designed to be the most challenging and entertaining PvE content available in the game.

n. **Loot:** Stuff (treasure: items or money) you get from mobs or containers (barrels, boxes, chests, etc).

o. **Equipment:** Equipment refers to any item that can be equipped in an equipment slot, as distinguished from items that can only be carried in inventory.

p. **Loot system:** A Loot System is any method to distribute items amongst a group of players. Whenever two or more players share some reward, they participate in a loot system, even if it's only the default roll system from Blizzard. Frequently, the terms DKP and "loot system" are used interchangeably, because DKP based systems are the most widely used type of formal loot systems.

q. **Boss:** Boss is a quite general phrase used for several special types of mobs. A common characteristic for all bosses is that they have a unique name and appear only once in the game. Bosses are harder to kill than the "normal" elite mob of the same level, and almost all bosses are immune to Crowd Control, though some may be susceptible in accordance with their intended mechanics

r. **Mobs:** A mob (short for mobile) is a generic term for any non-player entity whose primary purpose is to be killed for experience, quest objective, or loot.

s. **Instance:** Major hostile areas like dungeons, keeps, and other confined areas can have sub-areas called instances (aka instance(d) dungeons). These instances are special areas in the World of Warcraft where your group or raid party is able to interact with a dungeon privately; that is, without interference from other parties or raids.

t. **Quest:** A quest is a task given to a player character that yields a reward when completed. Most quests are given by an NPC (non-player character).

u. **Authentication**: The process of verifying the identity of a user, based on provided credentials.

v. **Authorization**: The process of determining if a user has proper permission to perform a particular action or read particular data, assuming that the user is successfully authenticated.

w. **Granted authority**: The permission of the authenticated user.

x. **Role**: A group of permissions of the authenticated user.

# Chapter 3 - Functional Requirements

## 8. Business Process Model

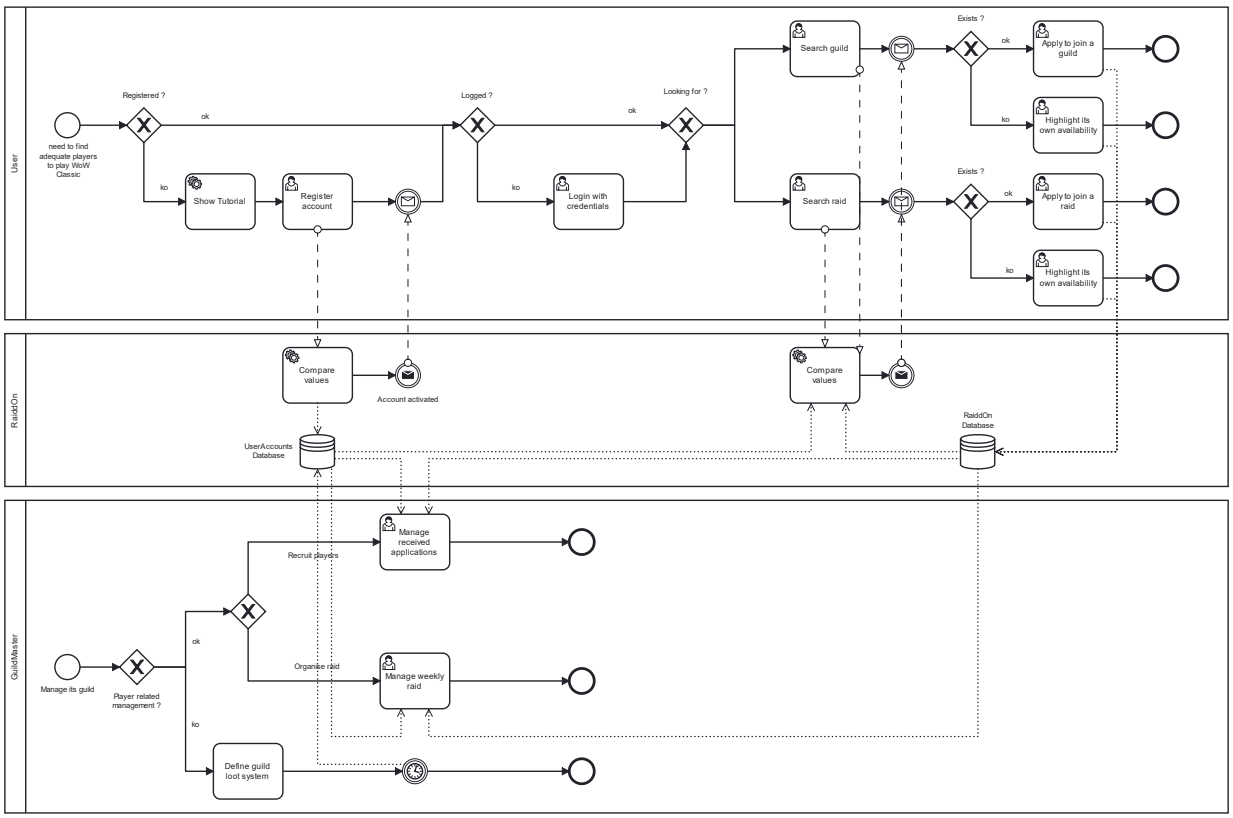



***Figure 3. Business Process Model for Raiddon.***

The business process model (Figure 3) represents the sequence of activities to be carried out by Raiddon users. The beginning of the forementioned sequence of activities is marked by the need of a World of Warcraft (hereafter, WoW) player to find other players sharing the same in-game objectives. Each player must have a Raiddon account and at least one playable character added to their account.

A logged-in user has access to a detailed guilds page in which they can search all registered guild on Raiddon. This ultimately allows the user to apply to join one of the guilds that satisfies his in-game ambitions. Alternatively, the same user can also search raids and therefore apply to join a raid if they are eligible to do so. It is to note that each application require approval from the guild master and the raid organizer respectively.

If the user is not admitted in the guild nor the raid or they do not find any results that match their needs they can do one of the following actions by updating their profile:

- Define their availability to raid,
- Define their availability to join a guild.

Another sequence of activities is marked by the need of a WoW guild master player to manage their guild which is challenged by two factors: one related to managing the guild's members (i.e., delete, downgrade or upgrade player) and one related to managing the guild's core value (i.e., guild loot system). A wow guild master is also faced by the obligation to provide weekly raids for their members. To do so they have to organise raids as well as manage received applications from players wishing to join the guild.

## 9. Use Cases (UC)

### 9.1 Guest management

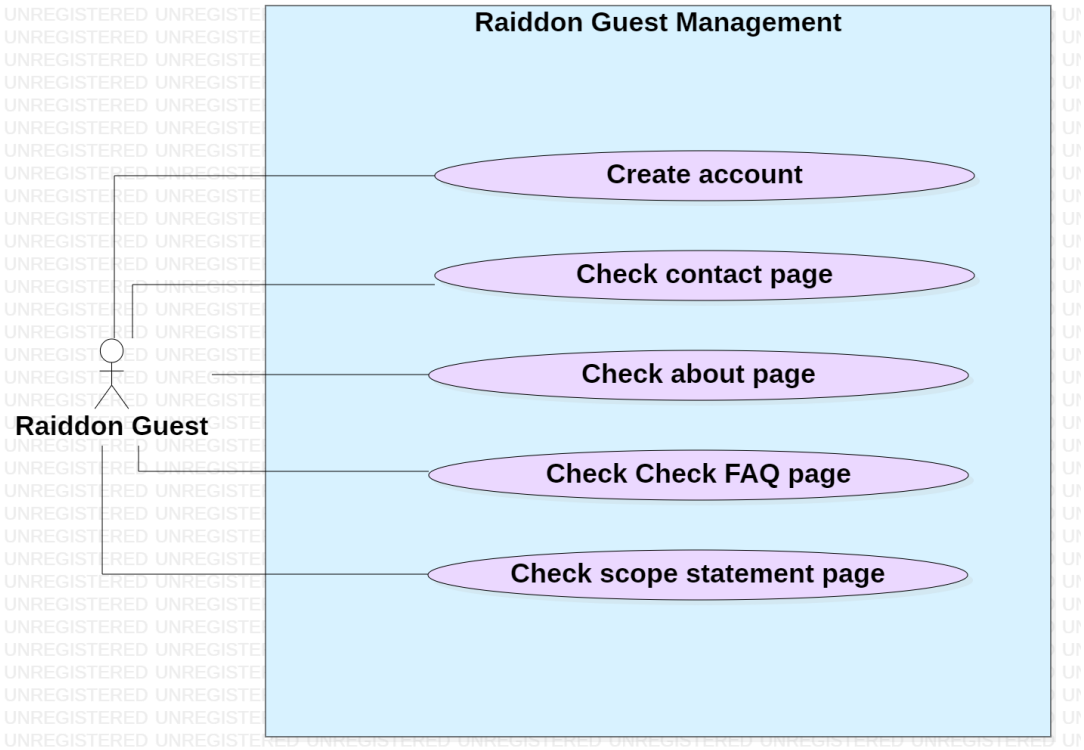



*Figure 4. All accessible activities for a Raiddon guest who does not have a Raiddon account yet.*

#### 9.1.1 Public pages

Public pages such as contact, FAQ, about and scope statement are accessible by all users whether they have an account or not. A detailed use case for each action will not be provided since the latter is the very straightforward. The user only has to click on the page name and the system will redirect them to the page in question.

- Contact page: allows the user to contact a Raiddon administrator.
- About page: a brief presentation of the website and its objectives.
- FAQ page: allows the user to find answers for questions they might have.
- Scope statement page: will redirect the user to a pdf copy of this document in order to give them more insight through the project's objectives and setup.

#### 9.1.2 Create account

| Use case name | Create Account |
|---|---|
| ID | CU-1 |
| Objective | Create an account on Raiddon website |
| Main Actors | Guest user |
| Secondary actors | N / A |
| Description | CU-1 begins when the main actor wishes to create an account. They create their username, password and fill in the requested information. CU-1 ends when the mandatory |

| | |
|---|---|
| | information is validated and registered in the management system |
| Pre-conditions | The main actor must not already have an account and the interface must be accessible. |
| Post-conditions | 1. The main actor changes status and now becomes a Raiddon user.<br>2. The user's account is registered in the organization's management system. |
| Scenario | 1. The main actor opens the signup page<br>2. The system displays the fields to be completed under a registration form:<br>• Username<br>• Password<br>• Password confirmation<br>• E-mail address<br>• Checkbox in order to accept terms & conditions<br>• The guest fills in and submit the form.<br>3. The system generates a standard profile picture for the profile<br>4. The system displays the home page of the logged in member |
| Exceptions | The account already exists. |
| Constraints | 1. CS1: The main actor must accept the terms of use.<br>2. CS2: All fields are mandatory. |
| Requirements | N / A |

### 9.2 User account management

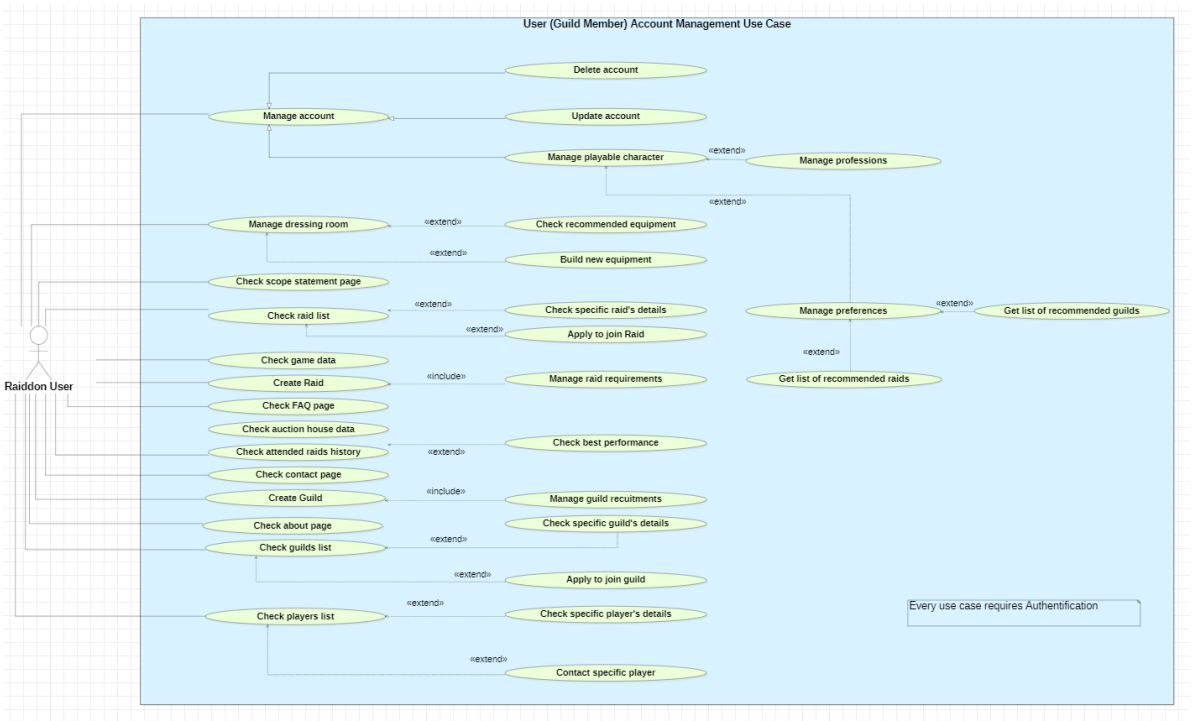



*Figure 5. Overview of all use cases that can be carried out by a user who is not a guild master or officer.*

### 9.2.1 Sign-in to account

| Use case name | Sign-in to account |
|---|---|
| ID | CU-2 |
| Objective | Sign in to Raiddon account |
| Main Actors | Raiddon registered user |
| Secondary actors | Raiddon management system |
| Description | CU2 begins when a Raiddon user wishes to access to their account and it ends when the Raiddon management system displays the account |
| Pre-conditions | The main actor must have an account |
| Post-conditions | 1. The main actor has access to their account and its functionalities again<br>2. In scenario 3 the management system is updated with the new password. |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor goes to their Profile page.<br>2. The main actor choses the login/sign-in option.<br>3. The system displays the required fields to fill in:<br>• Username<br>• Password<br>4. The main actor fills in the fields.<br>5. Credentials are recognised by Raiddon.<br>6. The system displays the actor's account |
| Scenario 2: Credentials are not recognised | 1. The main actor goes to the website homepage.<br>2. The main actor choses the login/sign-in option.<br>3. The system displays the required fields to fill in:<br>• Username<br>• Password<br>4. The main actor fills in the fields.<br>5. Credentials are not recognised by Raiddon.<br>6. The system displays an error message.<br>7. All fields are reset. |
| Scenario 3: The actor forgot their password | 1. The main actor goes to the website homepage.<br>2. The main actor choses the login/sign-in option.<br>3. The system displays the required fields to fill in:<br>• Username<br>• Password<br>4. The main actor chooses the option "I forgot my password".<br>5. The system displays the fields to fill in: User account email address.<br>6. The main actor fills in the field and validates.<br>7. The main actor receives a new temporary password by e-mail. |

| | |
|---|---|
| | 8. The main actor logs in with their new password.<br>9. The system displays the actor's account<br>10. The system informs the actor that they must change their temporary password shortly. |
| Exceptions | N / A |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-1 |

### 9.2.2 Manage account

| Use case name | Manage account |
|---|---|
| ID | CU-3 |
| Objective | 1. Define availability to join a guild<br>2. Define availability to participate to a raid<br>3. Delete Account<br>4. Update Account<br>5. Manage equipment<br>6. Manage professions<br>7. Manage reputations<br>8. Add new playable character<br>9. Manage playable characters |
| Main Actors | Raiddon registered user |
| Secondary actors | Raiddon management system |
| Description | CU-3 begins when a Raiddon user wishes to modify the information contained in their profile. The CU ends following the modifications. |
| Pre-conditions | The main actor must have an account and must be connected |
| Post-conditions | The main actor's personal information is updated. |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor goes to their Profile page.<br>2. The main actor selects "Edit my profile".<br>3. The system displays:<br>• Different pre-filled fields:<br>o Username<br>o E-mail address<br>• Playable character section:<br>o Character username<br>o Character race<br>o Character class<br>o Character faction<br>o Character specialisation<br>o Character Professions |

| | |
|---|---|
| | • A link allowing to upload a new profile picture.<br>4. The main actor edits one or more fields.<br>5. The main actor validates the modifications.<br>6. The system displays the modified profile. |
| Exceptions | 1. Ex1: If the information is syntactically invalid, the member must correct the fields in error.<br>2. Ex2: If the user has not filled in all the necessary fields, they must enter the information requested by the system. |
| Constraints | 1. CS3: All fields must be correctly filled in.<br>2. CS4: The profile image must be at least 200x200px, at most 400x400px, PNG/GIF/JPEG format and maximum size 2MB. |
| Requirements | N / A |
| Included UC | UC-2 |

### 9.2.3 Deleting account

| Use case name | Deleting account |
|---|---|
| ID | CU-4 |
| Objective | Deleting all information on Raiddon account |
| Main Actors | Raiddon registered user |
| Secondary actors | Raiddon management system |
| Description | CU-4 begins when a Raiddon user wishes to close their account and delete all their information from the website. To do this, they access this functionality from their profile. The CU ends when the user has finished his deletion request. |
| Pre-conditions | The main actor must have an account and must be connected |
| Post-conditions | The main actor's account is deleted |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor goes to their profile and validates the “delete my account” option.<br>2. The system displays:<br>• «Are you sure you want to delete your account? This action is irreversible, data such as raid history will no longer be available”<br>• A link “yes, I really want to delete my account” to validate the operation<br>• A link "no, I want to keep my account" invalidating and returning to the profile.<br>3. The main actor checks the appropriate box. |
| Exceptions | N / A |
| Constraints | CS5: The copy of the account present in the archive changes status and becomes inactive |

| | |
|---|---|
| Requirements | N/A |
| Included UC | UC-2 |

### 9.2.4 Manage dressing room

| | |
|---|---|
| Use case name | Manage dressing room |
| ID | CU-5 |
| Objective | Search all available items list in order to build an adequate equipment. |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-5 begins when a Raiddon user wishes to see the impact of certain items on their main playable character attributes(such as agility, intellect, etc.) |
| Pre-conditions | 1. The main actor must have an account and must be connected<br>2. The main actor must have at least added one playable character to their account. |
| Post-conditions | N / A |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor goes to their profile<br>2. The main actor choses the manage my equipment option from their dashboard<br>3. The system displays a list of items to fill in order to build the whole equipment pattern. It also shows a list of recommended items based on the main actor's main character's class.<br>4. The main actor choses each item slot and fills it with the adequate item.<br>5. The system displays all changes in the main actor's attribute.<br>6. |
| Exceptions | N / A |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-2, UC-3 |

### 9.2.5 Search available raid

| | |
|---|---|
| Use case name | Search available raid |
| ID | CU-6 |
| Objective | Search all available raid lists in order to join one |
| Main Actors | Raiddon registered user |

| | |
|---|---|
| Secondary actors | Raiddon user organising the raid |
| Description | CU-6 begins when a Raiddon user wishes to join a raid using the search raid tool. The CU ends when the user applies for a raid and waits the organising user response. |
| Pre-conditions | 1. The main actor must have an account and must be connected<br>2. The main actor must have at least added one playable character to their account. |
| Post-conditions | A “join raid” request is sent from the main actor of the CU to the secondary actor of the CU. |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor consults the available raids list<br>2. The main actor choses a raid that they wish to join<br>3. The system verifies that the actor satisfies all the required conditions in order to join the raid in question.<br>4. The system displays to the actor all the playable characters with which they are eligible to join the raid.<br>5. The main actor choses the playable character they wish to raid with.<br>6. The main actor now has a pending “join raid” application in their profile.<br>7. The raid organiser is notified and the “join raid” application is added to their “manage join raid applications” section. |
| Exceptions | N/A |
| Constraints | CS5: The copy of the account present in the archive changes status and becomes inactive |
| Requirements | N / A |
| Included UC | UC-2, UC-3 |

### 9.2.6 Check attended raids history

| | |
|---|---|
| Use case name | Check attended raids history |
| ID | CU-7 |
| Objective | Search all available raid lists in order to join one |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-7 begins when a Raiddon user wishes to check their past raids history. |
| Pre-conditions | 3. The actor must have an account and must be connected<br>4. The actor must have found at least one raid via Raiddon |
| Post-conditions | N / A |
| Scenarios | |

| | |
|---|---|
| Scenario 1: Nominal | 1. The main actor goes to the page corresponding to his raid history from their profile.<br>2. The system displays for all linked raids:<br>• Raid id<br>• Raid date<br>• Raid time<br>• Raid name<br>• Raid encounters<br>• A link towards the warcraft logs page. |
| Exceptions | N / A |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-2, UC-3 |

### 9.2.7 Organise a raid

| | |
|---|---|
| Use case name | Organise a raid |
| ID | CU-8 |
| Objective | The main actor wishes to organise a raid. They can define the raid name, date, time and duration. |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-8 begins when a Raiddon user wishes to organise a raid or an instance. It ends when the raid is published on the website |
| Pre-conditions | 1. The actor must have an account and must be connected<br>2. The actor must have found at least one playable character in their account. |
| Post-conditions | The raid is published on the website and added to the available raids list. |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor follows the steps of CU-2.<br>2. The main actor choses the organise raid option.<br>3. The system verifies that the main actor satisfies all the requirements in order to organise a raid.<br>4. The system displays the list of fields to fill in:<br>• Raid name<br>• Raid date and time<br>• Raid duration<br>• Raid loot system<br>• Raid requirements<br>• Comments<br>5. The main actor fills in the fields and submit the form.<br>6. The system displays: “Your raid has been published on the website” |

| | |
|---|---|
| Scenario 1:<br>The system does not validate the raid creation | 1. The main actor follows the steps of CU-2.<br>2. The main actor choses the organise raid option.<br>3. The system verifies that the main actor satisfies all the requirements in order to organise a raid.<br>4. The system displays the error message: “Please add a playable character to your account in order to organise a raid” |
| Exceptions | EX3: If the information entered is incorrect, the system displays an error message. |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-2, UC-3 |

### 9.2.8 Create a guild

| Use case name | Create a guild |
|---|---|
| ID | CU-9 |
| Objective | The main actor wishes to create a guild. They can define the guild name, type, loot system and emblem. |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-9 begins when a Raiddon user wishes to create a guild. It ends when the guild is added to the active guilds list on the website. |
| Pre-conditions | The actor must have an account and must be connected |
| Post-conditions | The guild is added to the active guilds list on the website. |
| Scenarios | |
| Scenario 1:<br>Nominal | 1. The main actor follows the steps of CU-2.<br>2. The main actor choses the create guild option.<br>3. The system verifies that the main actor satisfies all the requirements in order to organise a raid.<br>4. The system displays the list of fields to fill in:<br>• Guild name<br>• Guild gameplay type<br>• Guild raids per week<br>• Guild emblem<br>• Guild loot system<br>• Guild objectives<br>• Guild website<br>5. The main actor fills in the fields and submit the form.<br>6. The system displays: “Your request is pending”.<br>7. The system will check if the guild exists and if the main actor is the real guild master.<br>8. The system notifies the main actor that their guild is created and added to the active guilds list on the website. |

| | |
|---|---|
| Exceptions | EX3: If the information entered is incorrect, the system displays an error message. |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-2, UC-3 |

### 9.3 Guild master account management

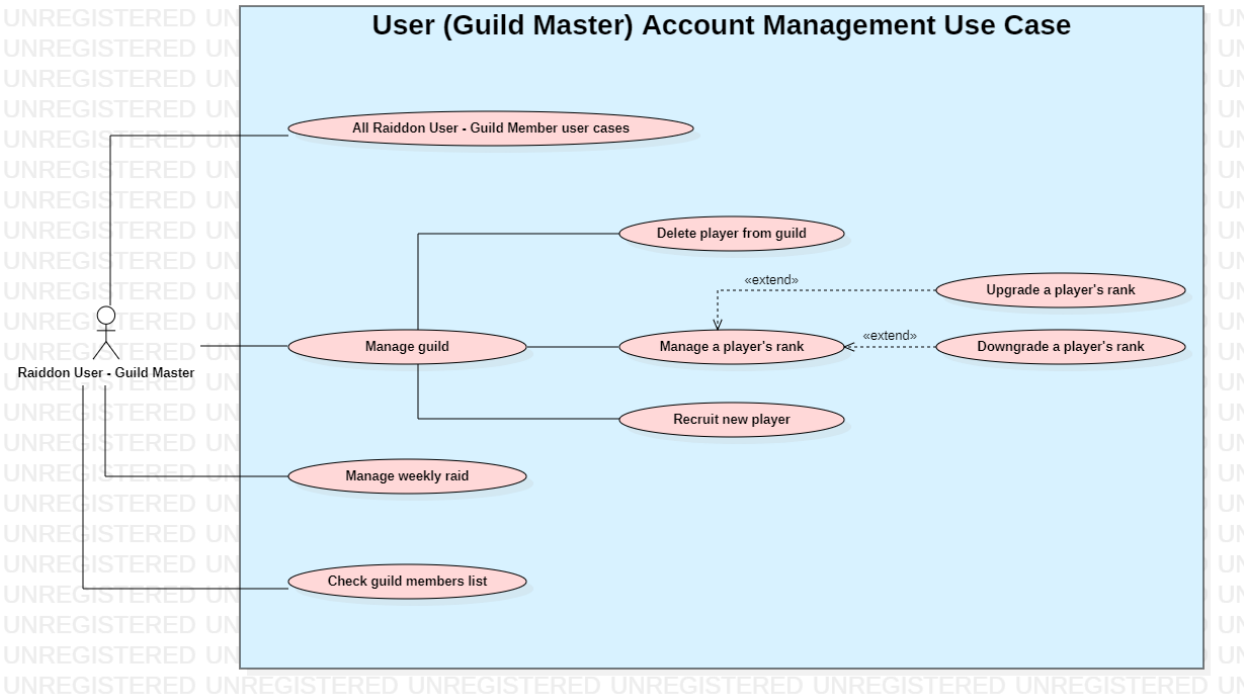



*Figure 6. Overview of all use cases that can be carried out by a user who is a guild master or officer.*

#### 9.3.1 Manage guild

| | |
|---|---|
| Use case name | Manage Guild |
| ID | CU-10 |
| Objective | 1. Upgrade a player's rank<br>2. Downgrade a player's rank<br>3. Delete a player from member list |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-10 begins when a Raiddon user wishes to manage their guild members. |
| Pre-conditions | 1. The main actor must have an account and must be connected.<br>2. The main actor must be a guild master. |
| Post-conditions | 1. The guild information is updated. |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor goes to their Profile page.<br>2. The main actor selects "Manage guild". |

| | |
|---|---|
| | 3. The system displays the following options:<br>4. Update guild details<br>5. Update guild member's list<br>6. The main actor choses "update guild details"<br>7. The system displays the following prefilled fields:<br>• Guild name<br>• Guild emblem<br>• Guild schedule<br>• Guild loot system<br>8. The main actor edits one or more fields.<br>9. The main actor validates the modifications. |
| Scenario 2:<br>The guild master choses to update members list | 1. The main actor goes to their Profile page.<br>2. The main actor selects "Manage guild".<br>3. The system displays the following options:<br>4. Update guild details<br>5. Update guild member's list<br>6. The main actor choses "update guild member's list"<br>7. The system displays a list of all members of the guild from which the main actor can select a member.<br>8. The main actor selects the member of their choice.<br>9. The system displays three options:<br>• Upgrade a player's rank<br>• Downgrade a player's rank<br>• Delete a player from member list<br>10. The main actor choses one of the three options.<br>11. The main actor validates the modifications. |
| Exceptions | 1. Ex1: If the information is syntactically invalid, the member must correct the fields in error.<br>2. Ex2: If the user has not filled in all the necessary fields, they must enter the information requested by the system. |
| Constraints | 1. CS3: All fields must be correctly filled in.<br>2. CS4: The guild emblem must be at least 200x200px, at most 400x400px, PNG/GIF/JPEG format and maximum size 2MB. |
| Requirements | N / A |
| Included UC | UC-2, UC-8 |

### 9.3.2 Check players available for raid list

| Use case name | Check players available for raid list |
|---|---|
| ID | CU-11 |
| Objective | To be able to consult all Raiddon users with the option “available for raid” turned on. |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-11 starts when the main actor wishes to check players available for raid. It ends when the system displays the list of players available for raid. |
| Pre-conditions | The main actor must have an account and must be connected. |
| Post-conditions | N / A |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor signs in (CU-2).<br>2. The main actor navigates to the “Players” page.<br>3. The main actor choses the option “Players available for raid”.<br>4. The system displays the list of all players available for raid. |
| Exceptions | N / A |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-2 |

### 9.3.3 Check players looking for guild list

| Use case name | Check players looking for guild list |
|---|---|
| ID | CU-12 |
| Objective | To be able to consult all Raiddon users with the option “looking for guild” turned on |
| Main Actors | Raiddon registered user |
| Secondary actors | N / A |
| Description | CU-11 starts when the main actor wishes to check players looking for a guild. It ends when the system displays the list of players looking for a guild. |
| Pre-conditions | The main actor must have an account and must be connected. |
| Post-conditions | N / A |
| Scenarios | |
| Scenario 1: Nominal | 1. The main actor signs in (CU-2).<br>2. The main actor navigates to the “Players” page. |

| | |
|---|---|
| | 3. The main actor choses the option “Players looking for guild”.<br>4. The system displays the list of all players looking for guild. |
| Exceptions | N / A |
| Constraints | N / A |
| Requirements | N / A |
| Included UC | UC-2 |

### 9.4 Raiddon Overview

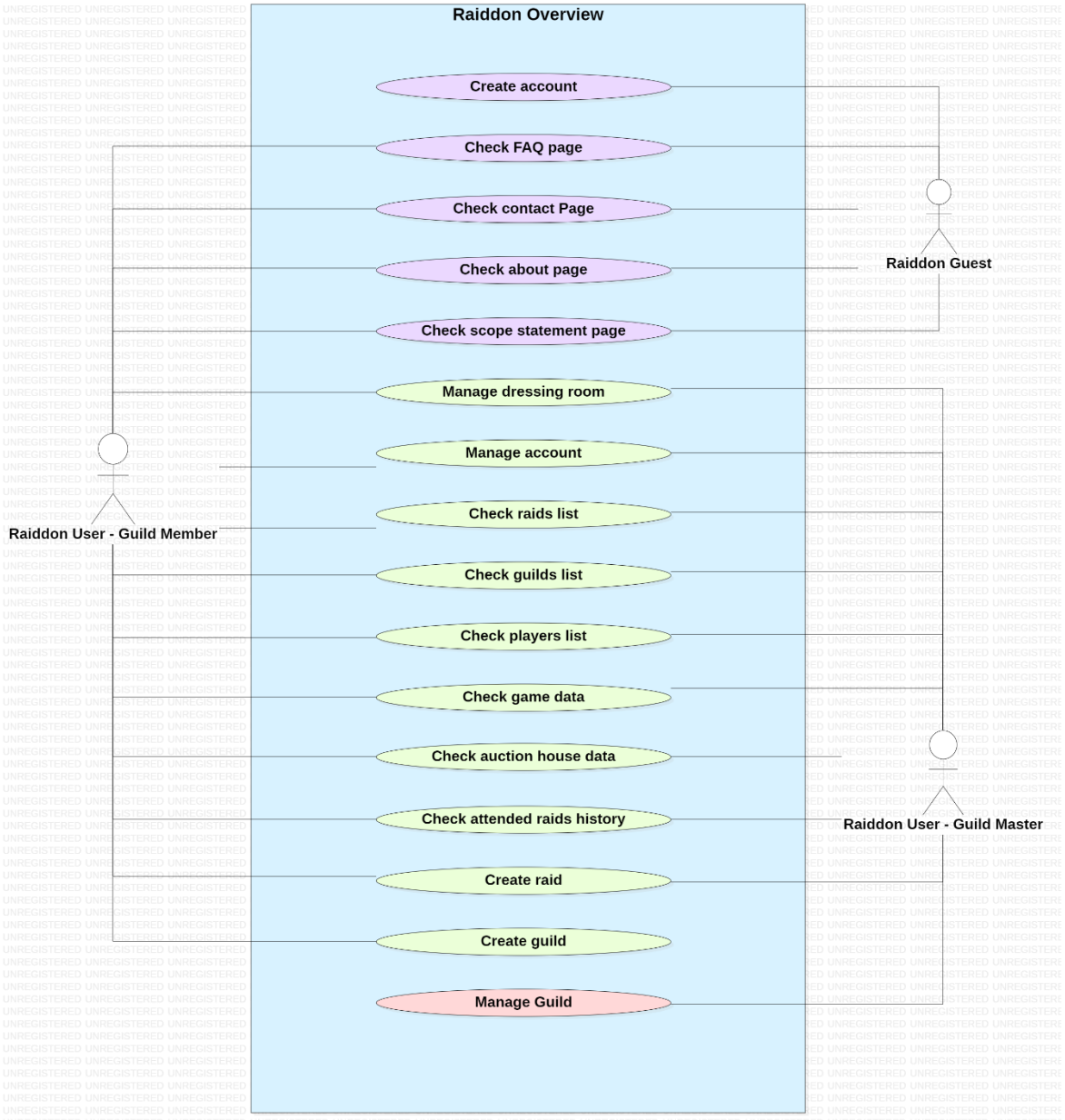



*Figure 7. Overview of all use cases that can be carried out using Raiddon. Green UCs represent all actions carried out by a user who is not a guild master while all the red ones represent actions carried out by a guild master or officer.*

## 10. Class diagram

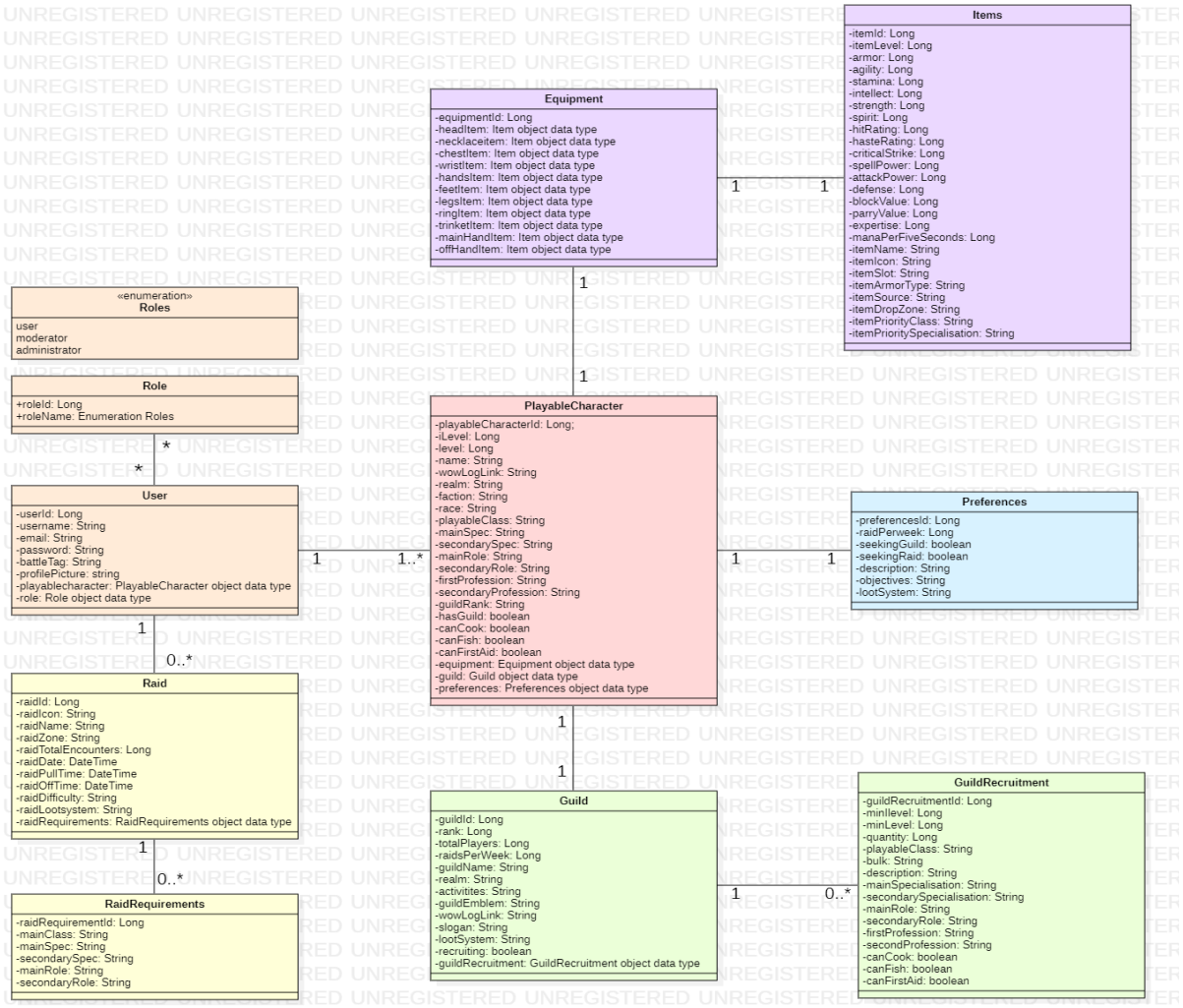



***Figure 8. Class diagram for Raiddon. Orange classes represent user details, yellow classes represent raid management, red class represents playable characters management, green classes represent guild management and purple classes represent equipment management.***

### 10.1 Class Diagram description

- **User**: Designates every person that will be using Raiddon website.
    - **Every user has**:
        - An identification number,
        - A username (not necessarily world of warcraft username),
        - An e-mail,
        - A password,
        - A battle tag (world of warcraft tag),
        - A profile picture,
        - Multiple playable characters,
        - A role.
        - Every user can be or not a raid leader, meaning when they create a raid they will automatically be considered as the leader of the created raid.

- Playable Character: Users must at least have one playable character in order to user the main functionalities of Raiddon.
    - Every playable character has:
        - An identification number,
        - An iLevel: intelligence level of the in-game character,
        - A level: level of the in-game character,
        - A name: must be the in-game name of the character,
        - A warcraft logs link,
        - A realm,
        - A faction,
        - A race,
        - A main class,
        - A main specialization,
        - A secondary specialization,
        - A main role,
        - A secondary role,
        - A first profession,
        - A secondary profession,
        - A guild or not,
        - A guild rank if they have a guild,
        - A set of items forming their equipment entity,
        - A set of preferences based on which the recommended guilds and raids will be filtered on the user profile.
- Guild: Playable characters of the same user can have or not a guild.
    - Every guild has:
        - An identification number,
        - A rank: the position of the guild in its own realm,
        - A total number of players,
        - A number of raids per week,
        - A name,
        - A realm,
        - A type of activities,
        - An emblem,
        - A warcraft logs link,
        - A slogan,
        - A loot system,
        - A set of recruitment preferences if the guild recruitment is open.
- Guild Requirements: Guilds that are open to recruitment have a set of recruitment options to choose from.
    - Every guild recruitment has:
        - An identification number,
        - A minimum iLevel of the person the guild wishes to recruit,

        - A minimum level of the person the guild wishes to recruit,
        - A quantity,
        - A playable class of the person the guild wishes to recruit,
        - A bulk of the recruitment,
        - A description,
        - A main specialisation of the person the guild wishes to recruit,
        - A secondary specialization of the person the guild wishes to recruit,
        - A main role of the person the guild wishes to recruit,
        - A secondary role of the person the guild wishes to recruit,
        - A first profession of the person the guild wishes to recruit,
        - A second profession of the person the guild wishes to recruit.
- Preferences: Every user can have or not preferences for each playable character defined in their profile.
    - Every preference has:
        - An identification number,
        - A number of raids per week,
        - An option to imply if the user is looking for a guild with the playable character in question,
        - An option to imply if the user is looking for raids with the playable character in question,
        - A description,
        - A set of objectives,
        - A loot system.
- Equipment: Every user can or not define the equipment of each playable character defined in their profile.
    - Every set of equipment has:
        - An identification number,
        - A head item,
        - A necklace item,
        - A chest item,
        - A wrist item,
        - A hands item,
        - A waist item,
        - A legs item,
        - A feet item,
        - A ring item,
        - A trinket item,
        - A main-hand item,
        - An off-hand item.
- Items: Items are what for a set of equipment of each playable character.
    - Every item has:
        - An identification number,

- A level,
- An armor value,
- An agility value,
- A stamina value,
- An intellect value,
- A strength value,
- A spirit value,
- A hit rating value,
- A haste rating value,
- A critical strike value,
- A spell power value,
- An attack power value,
- A defense value,
- A block value,
- A parry value,
- An expertise value,
- A mana per five seconds value,
- A name,
- An icon,
- A slot type,
- An armor type,
- A source,
- A drop zone,
- A priority class priority,
- A specialisation priority.

# Chapter 4 Data Fetching

## 11. Raiddon Architecture

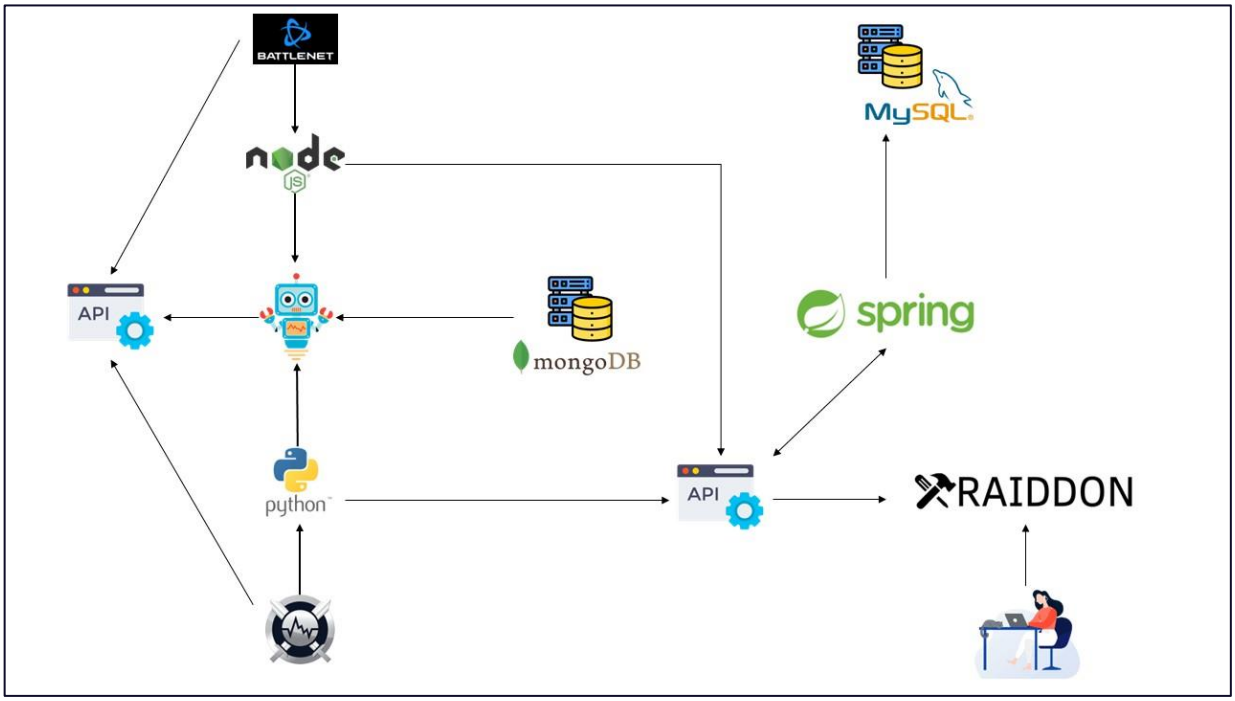



*Figure 9. Raiddon actual structure.*

As explained throughout this document, Raiddon follows a microservice architecture style that allows structuring the whole application as a collection of services (Figure 9). In the current project four microservices are developed using different technologies and languages:

a. **Raiddon battle net microservice:** https://raiddon-bnet-api.herokuapp.com/

This microservice is a flexible and feature-rich Javascript that allows fetching battle.net access token in order to access to Blizzard Battle.net APIs.

The code will first fetch your access token then fetch multiple sets of data from Battle.net and finally store it into a MongoDB database named "raiddon-bnet-api" The fetched data collections are:

- Achievements categories,
- Achievements,
- Areas,
- Armor,
- Auction house index,
- Alliance auction house,
- Horde auction house,
- Character specialisations,
- Class talents,
- Consumables,
- Containers,
- Creature families,
- Creature types,
- Gems,

- Item classes,
- Mounts,
- Playable classes,
- Playable races,
- Power types,
- Professions,
- Quest categories,
- Quest types,
- Realms,
- Reputation factions,
- Titles,
- Weapons.

b. **Raiddon warcraft logs microservice:** https://raiddon-wclog-api.herokuapp.com/
A sample OAuth2 Client Credentials flow application using Warcraft Logs APIs in Node.js and Python.
The app will fetch the access token, store it in a hidden JSON file and then fetch it in order to send a query to warcraft logs API. The fetch multiple sets of data from Battle.net and finally store it into a MongoDB database named "raiddon-wclog-api" The fetched data collections are:
   - Encounters,
   - Players Ranking,
   - Reports.

c. **Raiddon Spring Boot:** An important aspect for the acceptance of Service-Oriented Architectures is having convenient ways to help designers build secure applications. Spring security was used in this project in order to provide a secure user authentication. Further details can be found in the next chapter.

d. **Raiddon Angular framework:** The angular CLI version 11.2.2. was used to develop the front-end part of Raiddon. Further details can be found in next chapters.

**12. Fetching access tokens:**

Battle.net and Warcraft Logs use OAuth 2.0 for API authentication. OAuth allows clients to request and then use an access token to authenticate API requests ***(Jung et al., 2017)***. The first step in using OAuth is getting a "**client_id"** and "**client_secret"** by creating a new client on both websites.

The OAuth 2.0 Authorization Code Flow allows an application to access a user's data on their behalf ***(Rahman et al., 2020)***. This allows an application to acquire more sensitive, opt-in information about a user, such as a user's private reports ***(Singh et al., 2022)***, after obtaining the user's permission to do so. The Authorization Code Flow has two major parts: the authorization code request and the access token request.

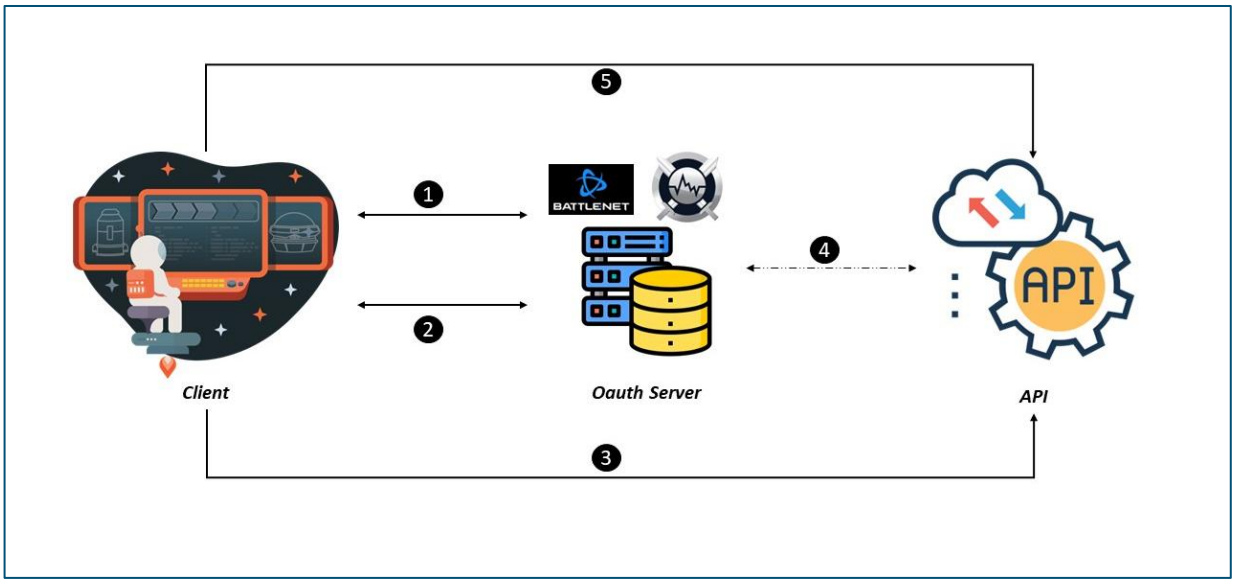



*Figure 10. OAuth 2.0 schema*

Access tokens last for 24 hours. A user changing their password, removing the authorization for an application's account, or getting their account locked for any reason, results in the expiration of their current access tokens. Developers should always check the response and request a new access token if the current one fails to work.

The authorisation flow adopted for fetching both databases is presented in Figure 10. The first step (1) consists of the client sending an authorisation request to the Oauth server that will return a token to the client (2). The access token is then sent from the client to the API service every time they perform a request for a protected resource access (3). The token is therefore checked by the resource server (4) in order to confirm that the client is authorised to consume the requested resource. Finally (5), the server responds with requested protected resources.

## 13. Fetching data from Battle.net

The World of Warcraft APIs provide developers with tools to enrich any fan site with information about World of Warcraft, StarCraft II, Diablo III, and Hearthstone. They can be integrated into websites in several ways:

- Add World of Warcraft character achievements, guild information, and Mythic Dungeon leaderboards.
- Display player profiles, ladder information, and match history for StarCraft II.
- Retrieve Diablo III player hero information, season leaderboards, and character class and skill information.
- Return information about Hearthstone cards and decks.

The following OAuth endpoints require an access token retrieved via the OAuth authorization code flow:

- GET /oauth/userinfo
- GET /profile/user/wow
- GET /profile/user/wow/protected-character/{realm-id}-{character-id}
- GET /profile/user/wow/collections
- GET /profile/user/wow/collections/pets
- GET /profile/user/wow/collections/mounts

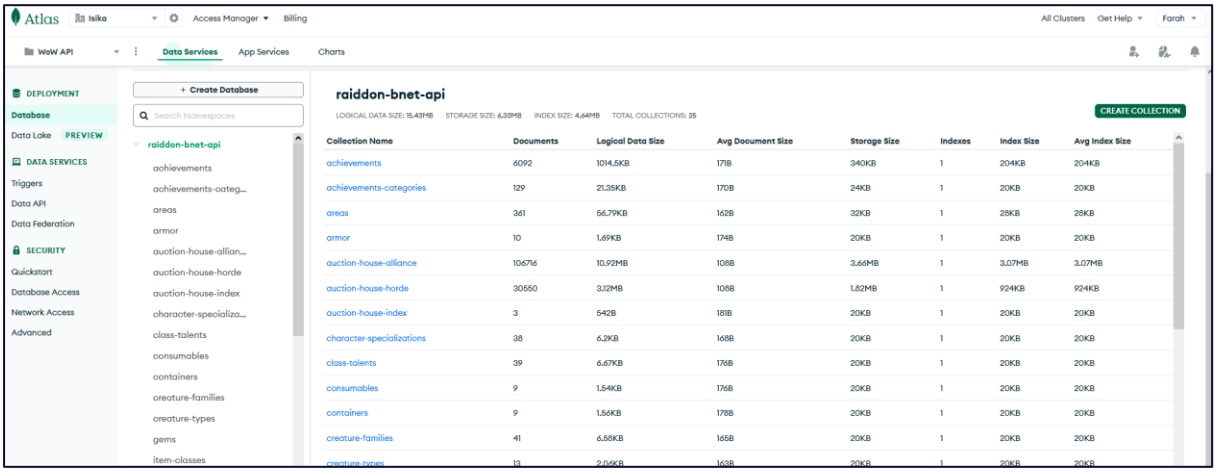



*Figure 11. Raiddon-Bnet-Api database in Mongo Atlas.*

## 14. Fetching data from warcraft logs

OAuth requires two commonly-used URIs to work properly. The authorize_uri gets user authorization to allow applications to access certain information, such as the fights and events of a private report. The token_uri submits items, such as authorization codes, to request access tokens applications can use with the v2 API.

- The Authorization URI is: https://www.warcraftlogs.com/oauth/authorize
- The Token URI is: https://www.warcraftlogs.com/oauth/token

The v2 APIs require access tokens granted one of three possible flows: the client credentials flow or the authorization code flow, or the PKCE code flow. The code used to fetch battle.net data in this project can be found on this GitHub Repository.

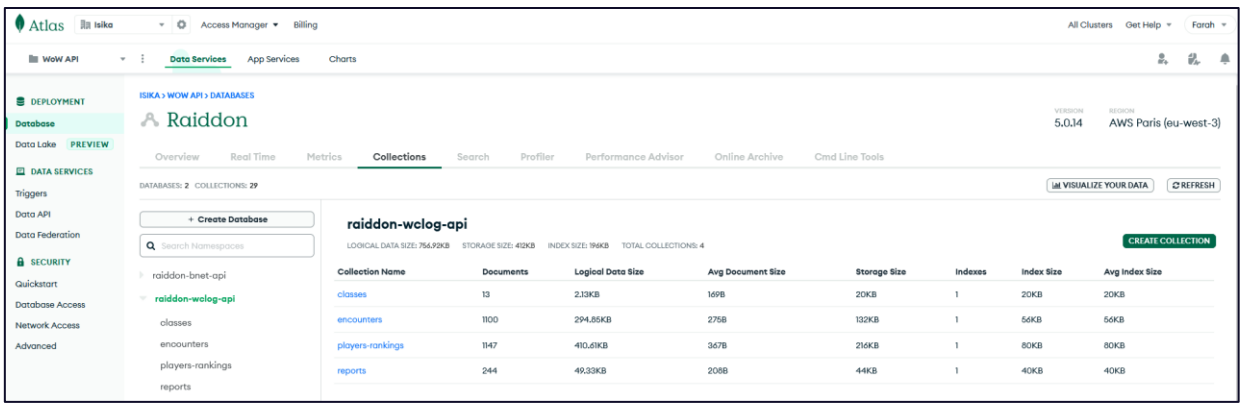
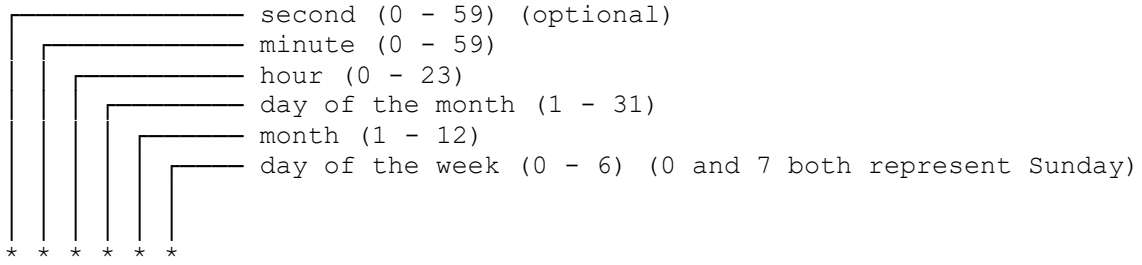



*Figure 12 Raiddon-Wclog-Api database in Mongo Atlas.*

**15. Scheduled Node-Cron Jobs**

Node Cron is a module that allows task scheduling in NodeJS. It is used within Raiddon in order to refresh the database every week and fetch new data from battle.net and warcraft logs.

The following expression is used to specify the schedule on which the Cron job should executed:




***"cron.schedule(* * * * * )"***

Each asterisk can be replaced according to the preferences of each project. For example, if the asterisk symbol is in the "month" field, it means the task is run every month.

# Chapter 5  Spring Boot Server

## 16. Spring Boot introduction

Spring Boot is a micro-framework that is available in an open-source format ***(Katamreddy et al., 2022)***. This framework is popular for providing an automated configurable Spring application to Java Developers ***(Soni et al., 2021)*** that simplifies their task of building complex applications. Java developers fancy this tool as it allows them to commence their work without any delay on the Spring application. The Spring Boot user community is massive and any new user can find plenty of resources and tutorials to get you started. This easy accessibility plays a key role in the ever-rising popularity of this framework ***(Rajput et al., 2018)***. has had a massive effect on the framework's popularity.

Spring Boot acts as a great choice for Application Development due to the following features:

- **Auto-Configuration:** Spring Boot provides you with an auto-configuration feature that makes more than 200+ decisions related to Application Development and automatically configures numerous functionalities by simply examining the dependencies ***(Modugu et al., 2020)***. It also helps you in detecting the occurrence of a Class in the Class path and then configures it automatically for you. The auto-configuration feature saves you a lot of work and reduces the time overhead for your development process.
- **Actuator:** The Spring Boot Actuator feature enables you to monitor what's going on inside a working Spring Boot application ***(Sadakath et al., 2018; Sharma et al., 2019)***. This feature is important as it prevents you from the risk of neglecting your application due to the auto-configuration feature. The Actuator offers great insight that can help you better understand your in-process Spring Boot applications.
- **Plug-ins:** Spring Boot offers a variety of plugins that you can apply to smoothly operate with embedded and in-memory databases ***(Sharma et al., 2019)***. Moreover, it facilitates the seamless creation and testing of Java applications via its default setup for unit and integration testing.

## 17. Spring Boot & dependencies withing Raiddon

https://github.com/Farah404/Raiddon-spring-server

Compared to other development frameworks, Spring Boot is known to be a revolutionary development paradigm that enables developers to build production ready application with far less development time and cost ***(Miao et al., 2020)***. Spring Boot starters are one of the key features of Spring Boot. They are mainly a set of pre-configured modules having common library dependencies; hence, freeing up the developer from searching a compatible version of libraries and configuring them manually. For example, spring-boot-starter-data-jpa starter module includes all the dependencies required to use Spring Data JPA ***(Reddy et al., 2017)***, along with Hibernate library dependencies, as Hibernate is the most commonly used JPA implementation.

Dependency injections is another fundamental aspect of the Spring framework that has been used throughout this project:

### 17.1 Spring dependencies:

#### 17.1.1 Spring-boot-starter-data-jpa: allows implementing JPA based repositories.

*<dependency>*

*<groupId>* ***org.springframework.boot*** *</groupId>*

```
        <artifactId> spring-boot-starter-data-jpa </artifactId>
    </dependency>
```

**17.1.2 Spring-boot-starter-security:** provides authentication, authorization, and protection against common attacks.

```
<dependency>
        <groupId> org.springframework.boot </groupId>
        <artifactId> spring-boot-starter-security </artifactId>
</dependency>
```

**17.1.3 Spring-boot-starter-validation:** allows validating user inputs.

```
<dependency>
        <groupId> org.springframework.boot  </groupId>
        <artifactId>  spring-boot-starter-validation </artifactId>
</ dependency >
```

**17.1.4 Spring-boot-starter-web:** allows building web RESTful applications using Spring MVC and Tomcat as a default embedded container.

```
<dependency>
        <groupId> org.springframework.boot </groupId>
        <artifactId> spring-boot-starter-web </artifactId>
</dependency>
```

**17.1.5 Spring-boot-starter-test and Spring-security-test:** contains majority elements for test. In the below dependencies, one thing to be noticed that it includes the scope of test <scope>test</scope>. It means when the application is bundled and packaged for deployment, any dependency that is declared with the test scopes is ignored. The test scope dependencies are only available when running in the development and Maven test modes.

```
<dependency>
        <groupId> org.springframework.boot </groupId>
        <artifactId> spring-boot-starter-test </artifactId>
        <scope> test </scope>
</dependency>
<dependency>
        <groupId> org.springframework.security </groupId>
        <artifactId> spring-security-test</artifactId>
        <scope> test </scope>
</dependency>
```

***17.1.6 Mysql-connector-java:*** allows using MySQL withing Raiddon project.

```
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        scope>runtime</scope>
</dependency>
```

## 18. Setting up the application properties

The ***application.properties*** file located inside the ***src/main/resources*** folder is used to define the application-related properties. It mainly contains the configuration to be followed which is required to run the application in a different environment. Properties like the server port and database connectivity can be found in this file.

A sample of the application/properties for Raiddon is given below:

```
# Angular Client
angular.url=${ANGULAR_URL}
# MySQL
spring.datasource.url=
jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT}/${MYSQL_DB}?serverTimezone=CE
T
spring.datasource.username=${MYSQL_USERNAME}
spring.datasource.password=${MYSQL_PASSWORD}
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Di
alect
# App Properties
raiddon.app.jwtCookieName= ${RAIDDON_COOKIE}
raiddon.app.jwtSecret= ${RAIDDON_SECRET}
raiddon.app.jwtExpirationMs= ${RAIDDON_EXPIRATION}
```

## 19. Project Packages

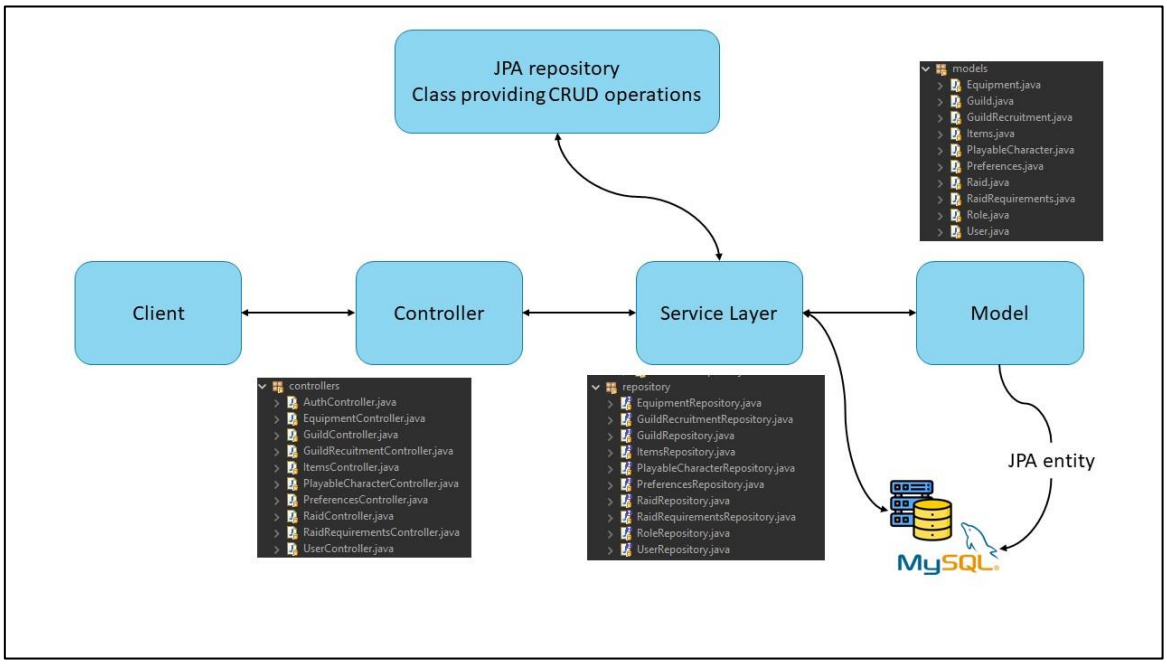



*Figure 13. Architecture flow of Raiddon.*

Spring boot uses all the features of Spring like Spring MVC, Spring Data and JPA ***(Mythily et al., 2022)***. A generic Spring Boot application consists of a controller which serves the clients HTTP request. This controller then interacts with the service layer which processes the request to modify the model and the database using the JPA repository through dependency injection. A

view page is returned to the user if no error occurred. The architecture of a typical Spring boot application is presented in Figure 10.

## 20. Token based Authentication with Spring Security & JWT (JSON Web Token)

Spring Security is an open-source Java framework, providing highly flexible and extensible authentication, authorisation, and access control solutions ***(Armando et al., 2014)***. Although it can be used for desktop applications, the main purpose of Spring Security is to secure web applications based on the Java Platform Enterprise Edition. Spring Security also provides support for basic role -based access control. Due to its flexible architecture the framework can easily be adapted and extended to support other forms of authentication and authorisation and access control as well ***(Xie et al., 2017)***.

JWT stands for JSON web token. It is commonly used for managing authorisations as well as creating a standard way between two parties that wish to communicate securely. JWT main purpose is authorisation which can be easily confused with authentication **(Jánoky et.al, 2018)** . An authentication process requires a username and a password in order to verify a login request. However, an authorisation process makes sure that the user sending the request to the server is the same user who is actually logged in during the authentication process. It mainly authorises the user to access to this particular authentication system **(Haekal et al., 2016)**.

Common authorisation process function based on session identification numbers (id). A session id is sent down in the cookies of the browser. Every time the client makes a request, they send the session id up to the server which in its turn checks its memory in order to find the user matching the received session id (Figure 14). JWT process is highly similar to the cookies process. However, instead of storing the user session's id in the server memory, the server will create a JWT that it encodes and serialises with its own secret key**(Jones et al., 2015)**. The generated token is then sent to the browser. The user now can make a request containing the JWT that will be verified by the server and hence is able to send the appropriate response (Figure 14).

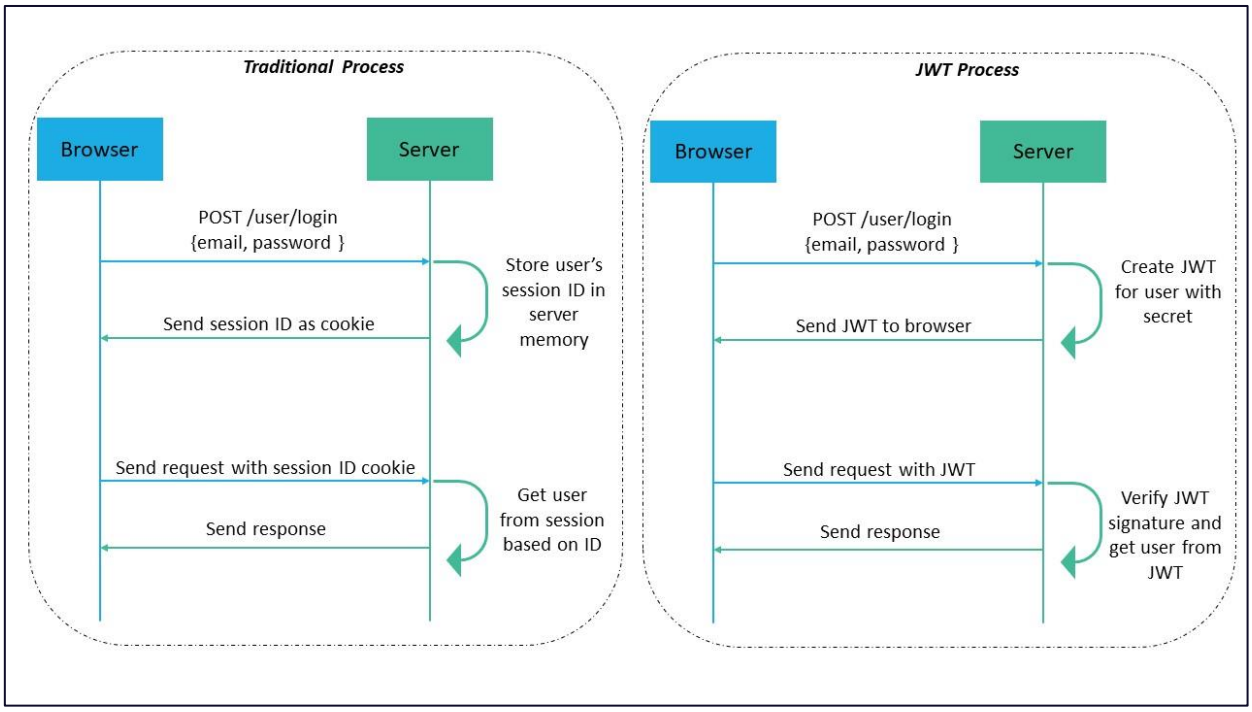



*Figure 14. Difference between traditional authorisation/authentication processes and the JWT process.*

In order to implement the forementioned framework into Raiddon, multiple criteria were respected **(Shingala et al., 2019)**:

- Appropriate flow for user signup and sign in using th JWT authentication was used.
- A specific spring boot application architecture was implemented.
- Spring security was configured in order to adapt it to JWT standards.

Figure 15 show an overview of the adopted work flow in order to build an application that supports token-based authentication with JWT.

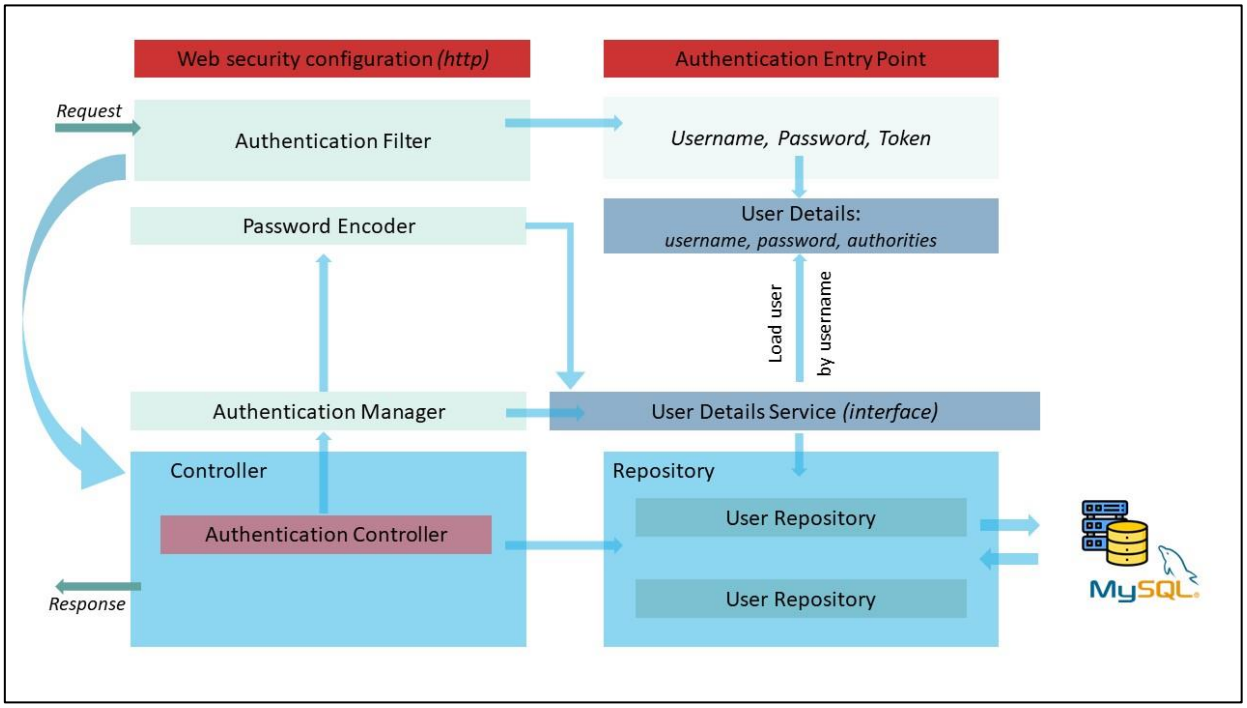



*Figure 15. Overview of the adopted work flow in order to build an application that supports token-based authentication with JWT.*

The core of this whole complex operation is the web security configuration **(Alex et al., 2004)** as it provides the Http security configurations allowing to setup cross-origin resource sharing (cors), cross-site request forgery (csrf), session management and rules for protected resources. In order to overcome the "most common" cors problem, a method has been implemented within Raiddon in order to allow requests from another domain outside the domain from which the first resource was served (Figure 16 & Figure 17).

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
  http.cors().and().csrf().disable()
      .exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
      .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
      .authorizeRequests().antMatchers("/api/auth/**").permitAll()
      .antMatchers("/api/test/**").permitAll()
      .anyRequest().authenticated();

  http.authenticationProvider(authenticationProvider());
  http.addFilterBefore(authenticationJwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);

  return http.build();
}
```

*Figure 16. Security filter chain bean used in the security configuration*

```java
@Bean
public CorsFilter corsFilter() {
        UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new UrlBasedCorsConfigurationSource();
        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.setAllowCredentials(true);
        corsConfiguration.setAllowedOrigins(Collections.singletonList("http://localhost:4200"));
        corsConfiguration.setAllowedHeaders(Arrays.asList("Origin", "Access-Control-Allow-Origin", "Content-Type",
                        "Accept", "Jwt-Token", "Authorization", "Origin, Accept", "X-Requested-With",
                        "Access-Control-Request-Method", "Access-Control-Request-Headers"));
        corsConfiguration.setExposedHeaders(Arrays.asList("Origin", "Content-Type", "Accept", "Jwt-Token", "Authorization",
                        "Access-Control-Allow-Origin", "Access-Control-Allow-Origin", "Access-Control-Allow-Credentials"));
        corsConfiguration.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"));
        urlBasedCorsConfigurationSource.registerCorsConfiguration("/**", corsConfiguration);
        return new CorsFilter(urlBasedCorsConfigurationSource);
}
```

*Figure 17. Cors filter bean*

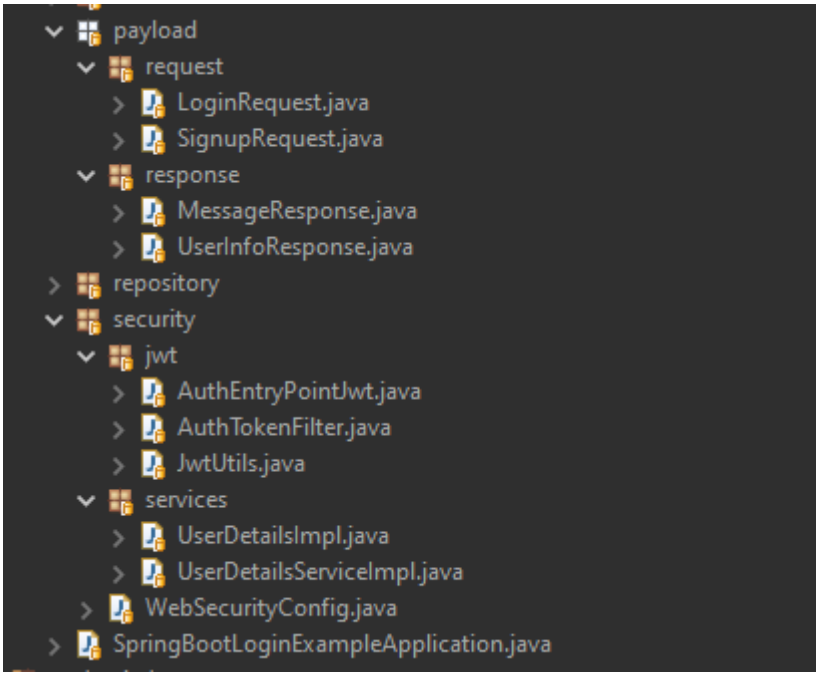



*Figure 18. Packages containing the JWT authorisation process*

After setting up the appropriate environment for the cors conditions, a "*UserDetails*" is created via the "*UserDetailsService*". "*UserDetails*" is a Spring Security object that is used for authentication and validation. It mainly contains the username, password and authorities necessary to build an authentication object. On the other hand, any errors will be intercepted by the authentication entry point.

The authentication filter handles the part where requests are being made to the API. It extends the "*OncePerRequestFilter*" providing the necessary methods to parse and validate the JWT as well as load the user details using "*UserDetailsService*" and check the authorization conditions using the "*UsernamePasswordAuthenticationToken*". The latter gets the username and password from the login Request (managed by the Authentication controller) and is validated by the authentication manager, the "*UserDetailsService*" and the password encoder.

The architecture of the packages containing all these java files can be found in Figure 18.

# Chapter 6 Angular Framework

## 21. Angular framework introduction

Angular is a TypeScript rich framework based on free and open-source web application framework **(Moiseev er al., 2018)**. The architecture of an Angular application relies on different fundamental concepts (Figure 19). An angular application is a set of several components that are organised into NgModules and that are used to defin the application's views **(Novac et al., 2021)**. Functionnalities are provided by the services section which are ultimately injected into the components as dependencies; making the code significantly more efficient.

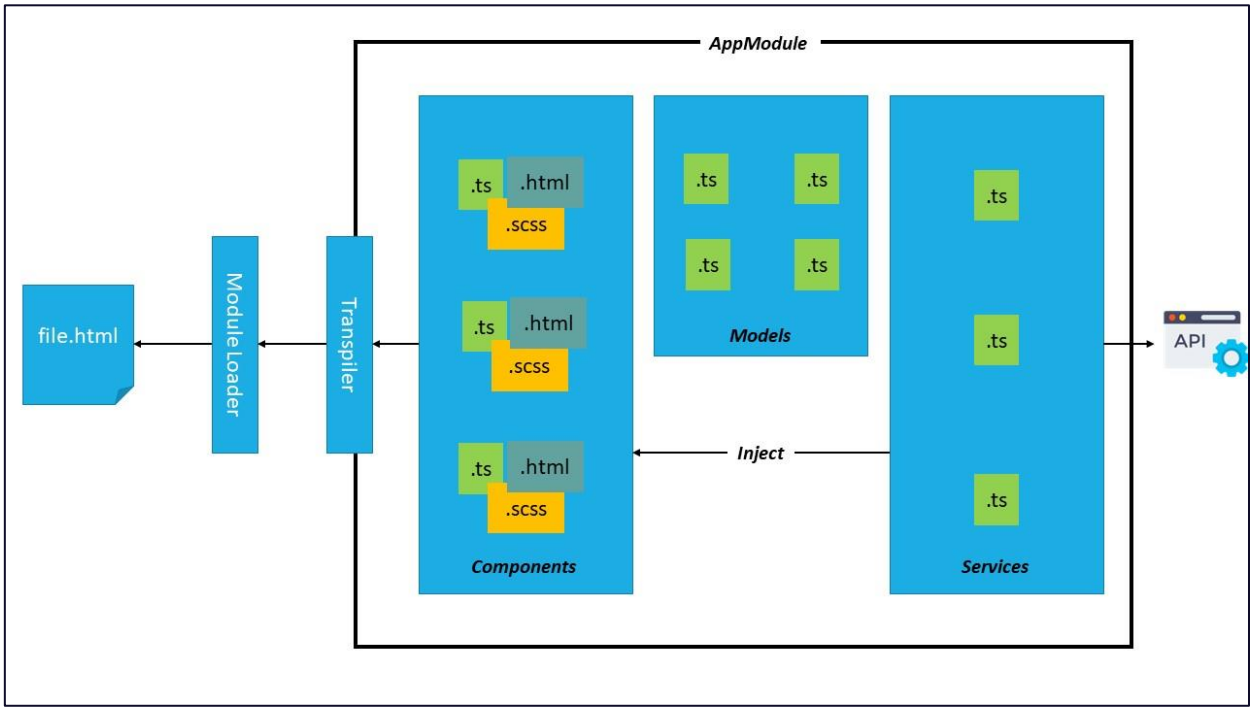



*Figure 19. Overview of a common Angular application.*

Components are then used to build the angular application. Every component consists of a TypeScript file, a html file and a styling scss file. To create a componnent the following command must be ran: *"ng generate component <component-name>"*

This command will generate the following:

- A directory named after the component
- A component file, <component-name>.component.ts
- A template file, <component-name>.component.html
- A CSS file, <component-name>.component.css
- A testing specification file, <component-name>.component.spec.ts

A component must belong to an NgModule in order for it to be available to another component or application **(Escott et al., 2019)**. In Figure 20, the assembly of different component in order to build a sample view is shown.

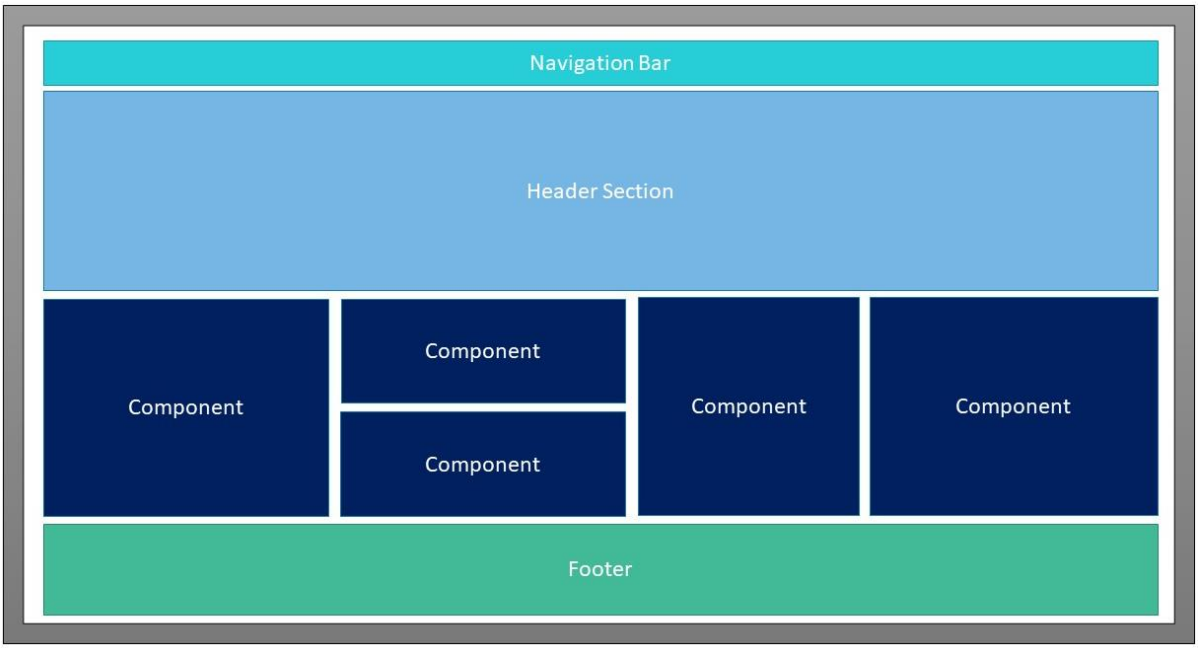



*Figure 20. Several components assembled together in order to form a sample view.*

## 22. Angular development dependencies within Raiddon

https://github.com/Farah404/Raiddon-angularFramework

The main dependencies used within Raiddon project are the following:

- "@angular/animations": "^14.2.3",
- "@angular/cdk": "^14.2.2",
- "@angular/common": "^14.2.3",
- "@angular/compiler": "^14.2.3",
- "@angular/core": "^14.2.3",
- "@angular/forms": "^14.2.3",
- "@angular/platform-browser": "^14.2.3",
- "@angular/platform-browser-dynamic": "^14.2.3",
- "@angular/router": "^14.2.3",
- "@auth0/angular-jwt": "^5.1.0",
- "@fortawesome/fontawesome-free": "^6.0.0",
- "@google/model-viewer": "^2.1.1",
- "angular-notifier": "^12.0.0",
- "rxjs": "~7.5.0".

## 23. Authentication within Raiddon via Angular

After implementing the JWT method on the Spring Boot part of the project, it now needs to be taken into account within the Angular framework. Figure 21 shows the detailed work flow in order to carry out the forementionned process which is mainly conducted by the services packages: the storage service, the authentication service and the user service. The storage service will provide the application component and the registration and login forms with the user token and information stored in the browser. The authentication service will send the

registration andlogin requests using Angular Http client. The latter will inspect every HTTP request before sending it to the back-end part of the applciation.

Therefore the Raiddon user will be ultimately allowed to register a new account as well as login to an existing account.

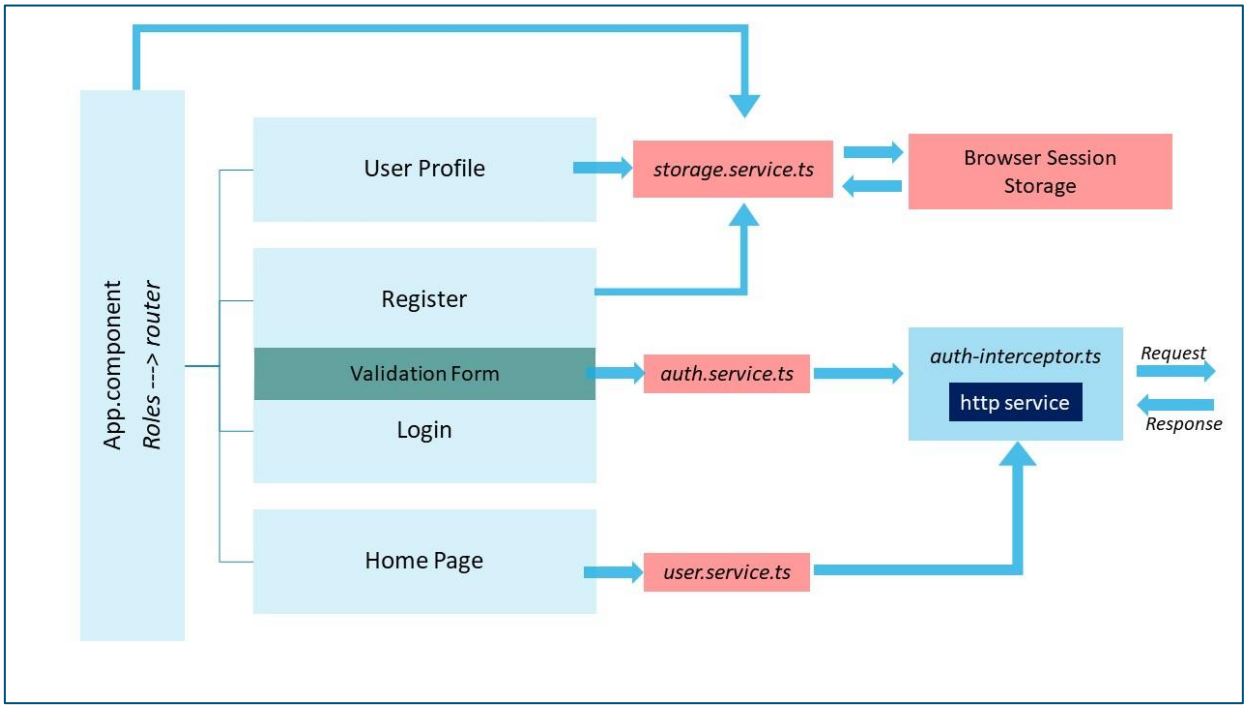



*Figure 21. Angular application diagram with router and HTTP Interceptor*

# Chapter 7 The Website

## 24. Home page

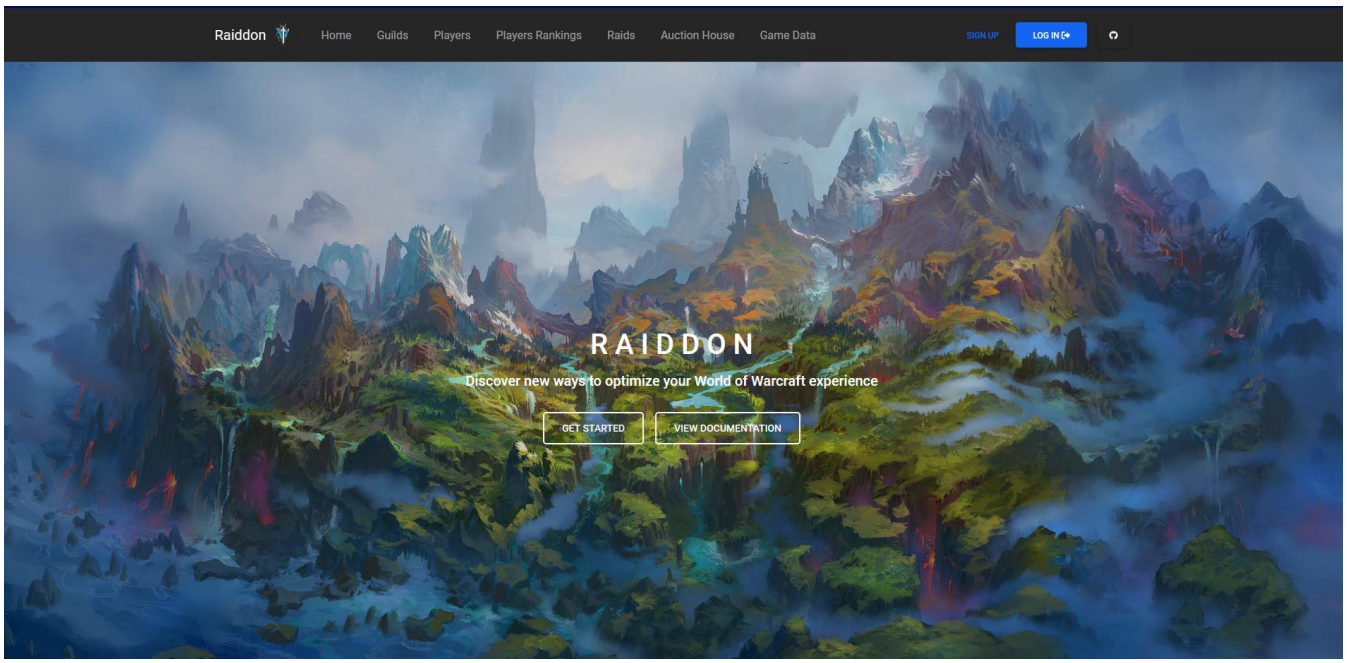



*Figure 22. Raiddon home page.*

## 25. Signup page

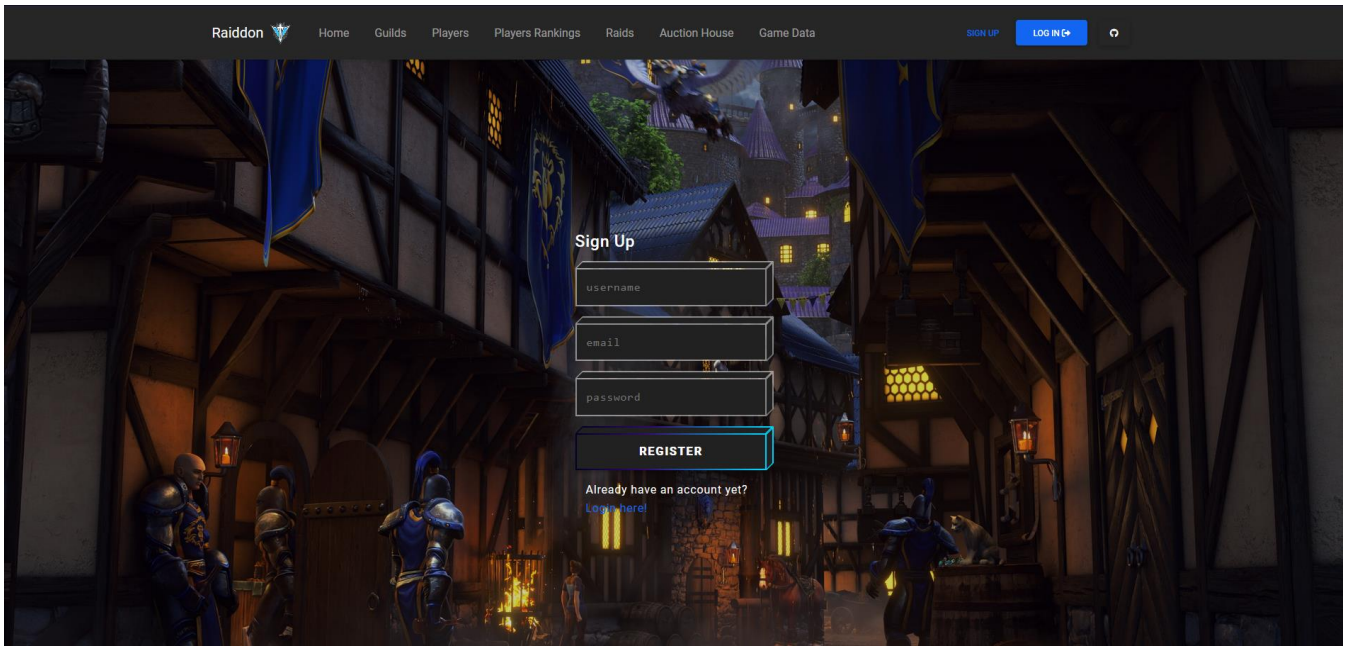



*Figure 23. Raiddon signup page.*

## 26. Login page

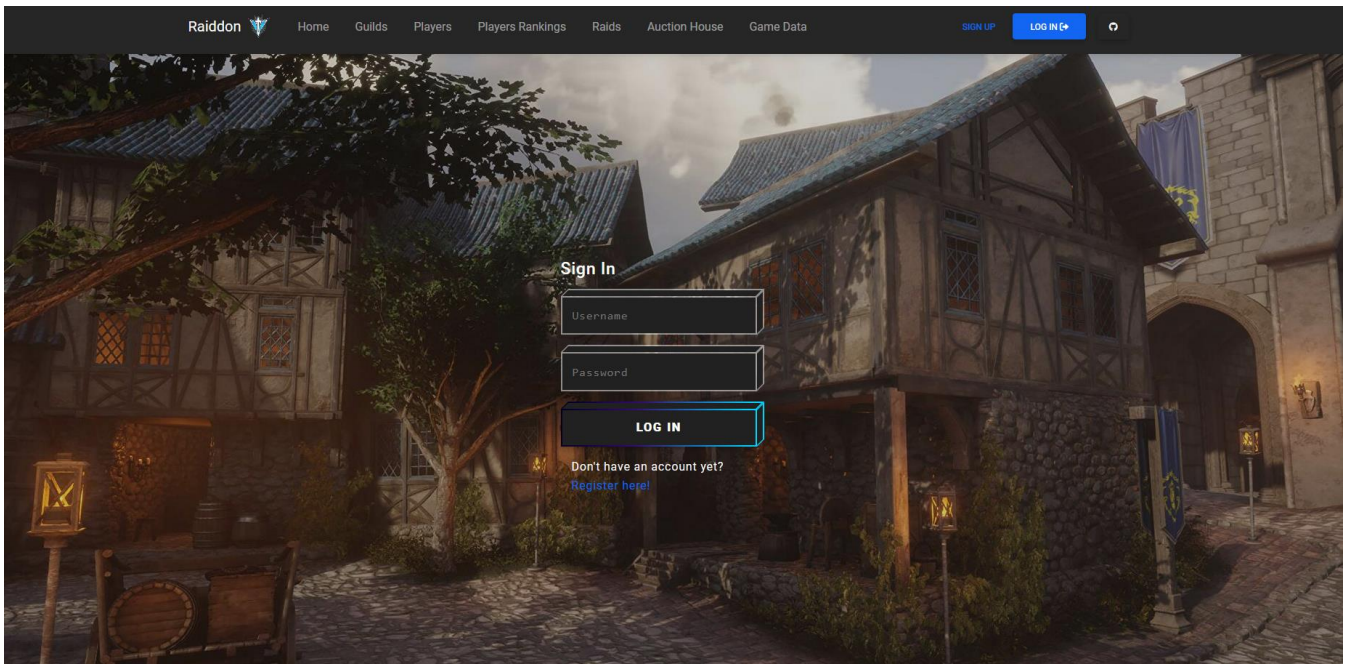



*Figure 24. Raiddon login page.*

## 27. Contact page

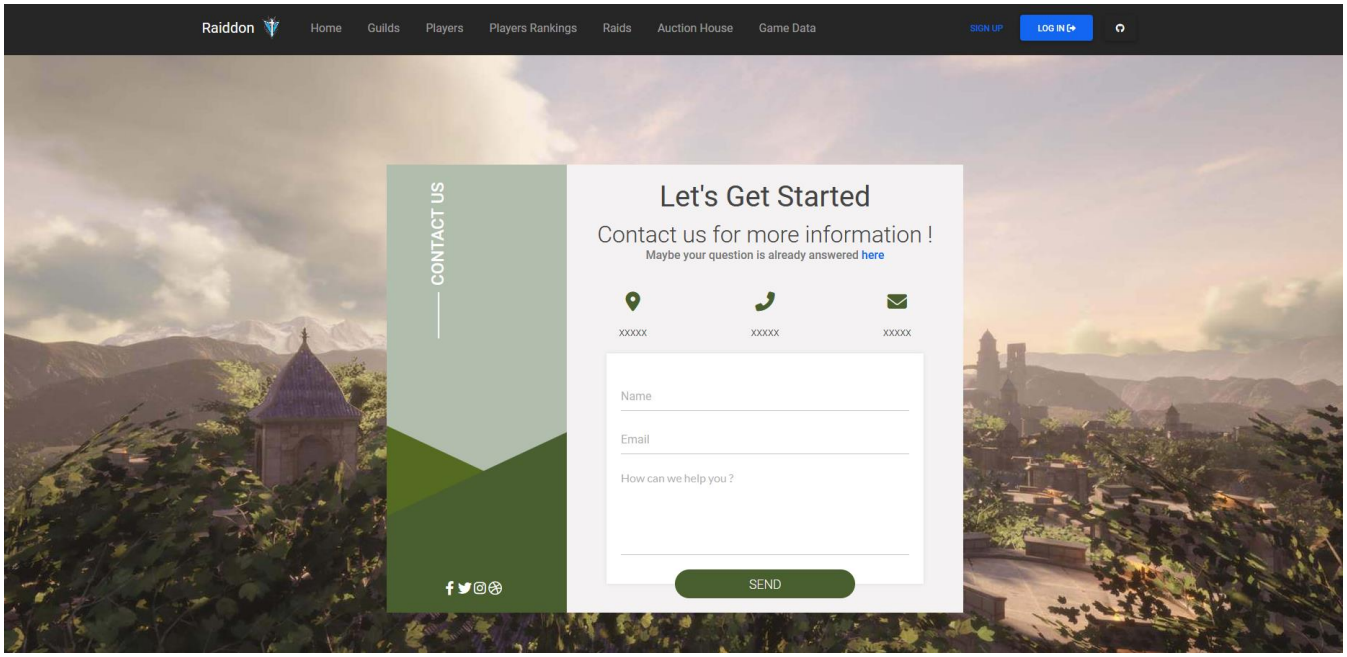



*Figure 25.Raiddon Contact page.*

# Chapter 8 Conclusion

Raiddon is a project that fosters all the developpement and conceptual understandings acquired throughout the ISIKA 10-month educational program.

The project respected the imposed microservices architecture by containing four independent microservices:

1- A NodeJS rich script microservice fetching data from battle.net API:
https://github.com/Farah404/Raiddon-bnet-data
https://raiddon-bnet-api.herokuapp.com/

2- A Python rich script microservice fetching data from warcraft logs API:
https://github.com/Farah404/Raiddon-wclog-data
https://raiddon-wclog-api.herokuapp.com/

3- A Spring boot microservice implementing JWT authorisation and handling the client management part:
https://github.com/Farah404/Raiddon-spring-server

4- An Angular framework microservice handling the front-end part of the project:
https://github.com/Farah404/Raiddon-angularFramework

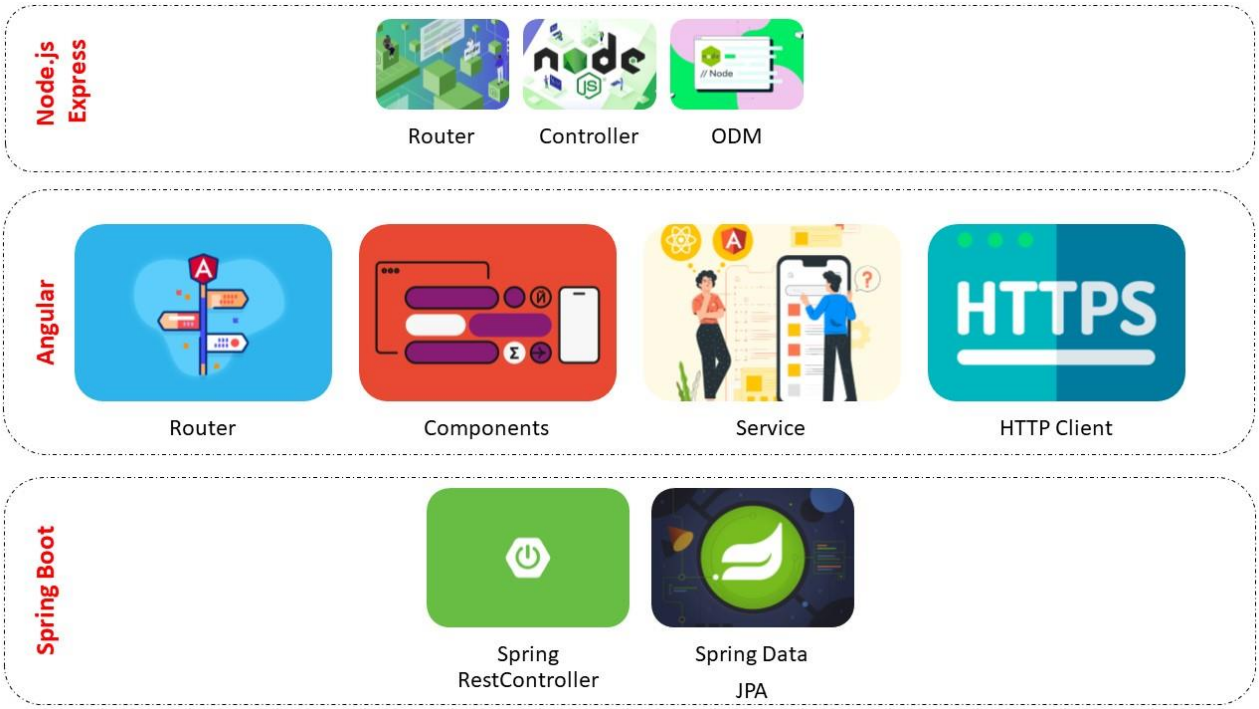



*Figure 26. Raiddon microservices overview*

## 28. References


Jung, S.W. and Jung, S., 2017. Personal OAuth authorization server and push OAuth for Internet of Things. *International Journal of Distributed Sensor Networks*, *13*(6), p.1550147717712627.

Rahman, S.S., Hossain, N., Hossain, M.A., Hossain, M.Z. and Sohag, M.H.I., 2020. OAuth 2.0: A Framework to Secure the OAuth-Based Service for Packaged Web Application. In Innovative Perspectives on Interactive Communication Systems and Technologies (pp. 92-139). IGI Global.

Katamreddy, S.P.R. and Upadhyayula, S.S., 2023. Spring Boot Essentials. In *Beginning Spring Boot 3* (pp. 47-55). Apress, Berkeley, CA.

Soni, R.K., Ganeshan, A. and Rajesh, R.V., 2017. *Spring: Developing Java Applications for the Enterprise*. Packt Publishing Ltd.

Rajput D. Mastering Spring Boot 2.0: Build modern, cloud-native, and distributed systems using Spring Boot. Packt Publishing Ltd; 2018 May 31.

Modugu SR, Farhat H. Implementation of the internet of things application based on spring boot microservices and REST architecture. InProceedings of the Computational Methods in Systems and Software 2020 Oct 14 (pp. 20-31). Springer, Cham.

Sharma S. Mastering microservices with java: Build enterprise microservices with Spring Boot 2.0, Spring Cloud, and Angular. Packt Publishing Ltd; 2019 Feb 26.

Sadakath MS. Spring Boot 2.0 Projects: Build production-grade reactive applications and microservices with Spring Boot. Packt Publishing Ltd; 2018 Jul 30.

Miao K, Li J, Hong W, Chen M. A microservice-based big data analysis platform for online educational applications. Scientific Programming. 2020 Jun 3;2020.

Reddy P, Siva K. Introduction to Spring Boot. InBeginning Spring Boot 2 2017 (pp. 1-20). Apress, Berkeley, CA.

Mythily, M., Raj, A.S.A. and Joseph, I.T., 2022, July. An Analysis of the Significance of Spring Boot in The Market. In *2022 International Conference on Inventive Computation Technologies (ICICT)* (pp. 1277-1281). IEEE.

Jánoky, L.V., Levendovszky, J. and Ekler, P., 2018. An analysis on the revoking mechanisms for JSON Web Tokens. *International Journal of Distributed Sensor Networks*, 14(9), p.1550147718801535.

Jones, M., Campbell, B. and Mortimore, C., 2015. *Json web token (jwt) profile for oauth 2.0 client authentication and authorization grants* (No. rfc7523).

Haekal, M., 2016, October. Token-based authentication using JSON web token on SIKASIR RESTful web service. In *2016 International Conference on Informatics and Computing (ICIC)* (pp. 175-179). IEEE.

Alex, B., Taylor, L., Winch, R., Hillert, G., Grandja, J. and Bryant, J., 2004. Spring Security Reference. *URL https://docs. spring. io/springsecurity/site/docs/current/reference/htmlsingle/.[utoljára megtekintve: 2017. 04. 21.], 12*.

Moiseev, A. and Fain, Y., 2018. *Angular Development with TypeScript*. Simon and Schuster.
Novac, O.C., Madar, D.E., Novac, C.M., Bujdosó, G., Oproescu, M. and Gal, T., 2021, June. Comparative study of some applications made in the Angular and Vue. js frameworks. In *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)* (pp. 1-4). IEEE.

Escott, K.R. and Noble, J., 2019, July. Design patterns for angular hotdraw. In *Proceedings of the 24th European Conference on Pattern Languages of Programs* (pp. 1-13).