Design Document

D&D Character Creator

Marijn Colen

459595

Contents

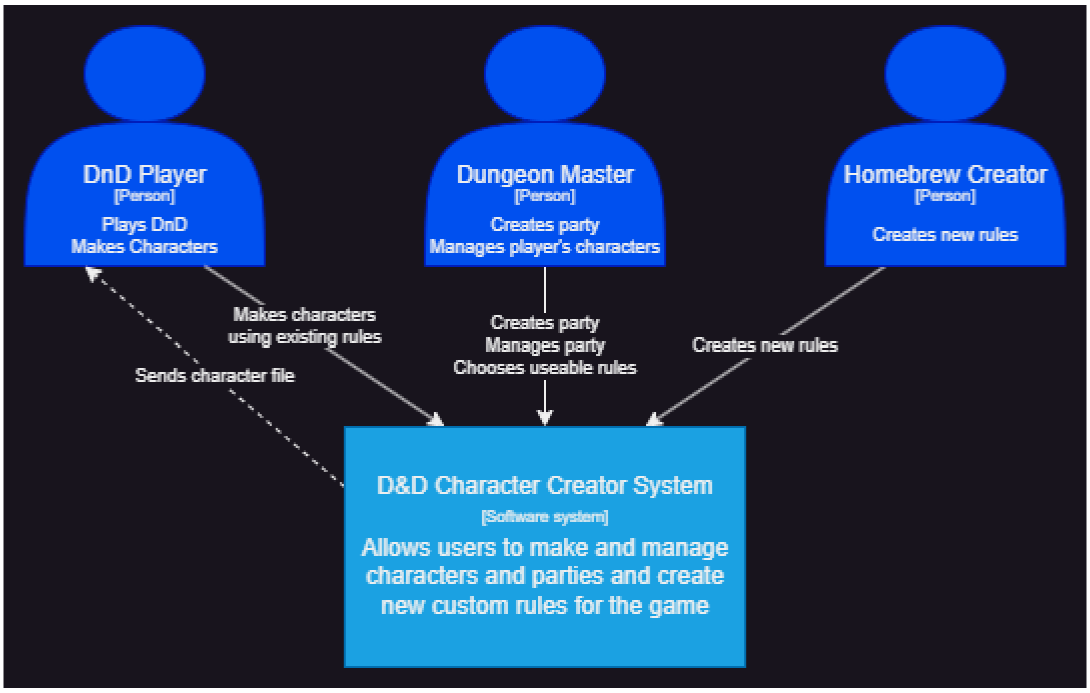
Important Context

What is Dungeons & Dragons?

Dungeons & Dragons, also known as D&D or DnD, is a tabletop roleplaying game that people can play with their friends. In D&D, a story is told over multiple “sessions”. A group of people playing D&D almost always consists of 1 Dungeon Master and one or more players. The **Dungeon Master**, or DM, is the storyteller. They are an impartial entity and don’t actively partake in the story themselves. The DM controls the monsters the heroes encounter and the people they interact with. Each **player** has their own character. This character is what they control while playing the game. Typically, every player will have only one character to control, but there are cases in which a player will control multiple characters at the same time. When there are multiple characters in a single D&D story, they are known as the **party**. The DM will start by telling a story and giving the players a scenario. The players then respond to this scenario as their characters. An important part of D&D is roleplay. This means that instead of responding to the scenario how they would personally respond to it, they must try to channel the personality of the character they created when making decisions. Each **character** has their own personality, skills and abilities. A character’s skills and abilities are decided by two major factors. The character’s race and class. The character’s **race** is the type of creature they are. Examples of this are Elves, Dwarves, Gnomes, or simply Humans. Some races, like Elves, have **subraces**. Some races, like Humans, don’t. This means the race is divided into several types that all fall under the general term of the race. For example, there are High Elves, Wood Elves, and Dark Elves. A character’s **class** is similar to a job in real life. For example, a character can be a Wizard, a Druid, a Fighter, or a Paladin. While a character’s race only gives them a few abilities, the character’s class is the main source they get their abilities from. Contrary to races, every class has multiple **subclasses**. This subclass gives them even more abilities and allows them to specialize in a specific type of magic, or a specific fighting style. For example, the Fighter class has an Arcane Archer subclass that specializes in using magic arrows, and a Battle Master subclass that specializes in using special techniques in close-quarters combat. Where races get their subrace (if they have one) at the very start of the game, some classes have to reach level 2 or 3 before they get to choose a subclass. At (almost) every **level**, a character will gain new abilities or get improvements to existing abilities. The Dungeon Master decides when the characters level up. When creating a character, you have to choose a race and a class, and a subrace if your chosen race has any. If your chosen class gets to choose their subclass at level 1, you have to choose the subclass immediately as well. For these races and classes, there are written rules from both official and unofficial sources. Rules that are written by unofficial sources, are called **homebrew rules**, or simply homebrew. Some people write custom rules for D&D in their spare time, and some people have even managed to make a living from creating and selling books with custom rules in them. These people are **homebrew creators**.

C4 diagrams

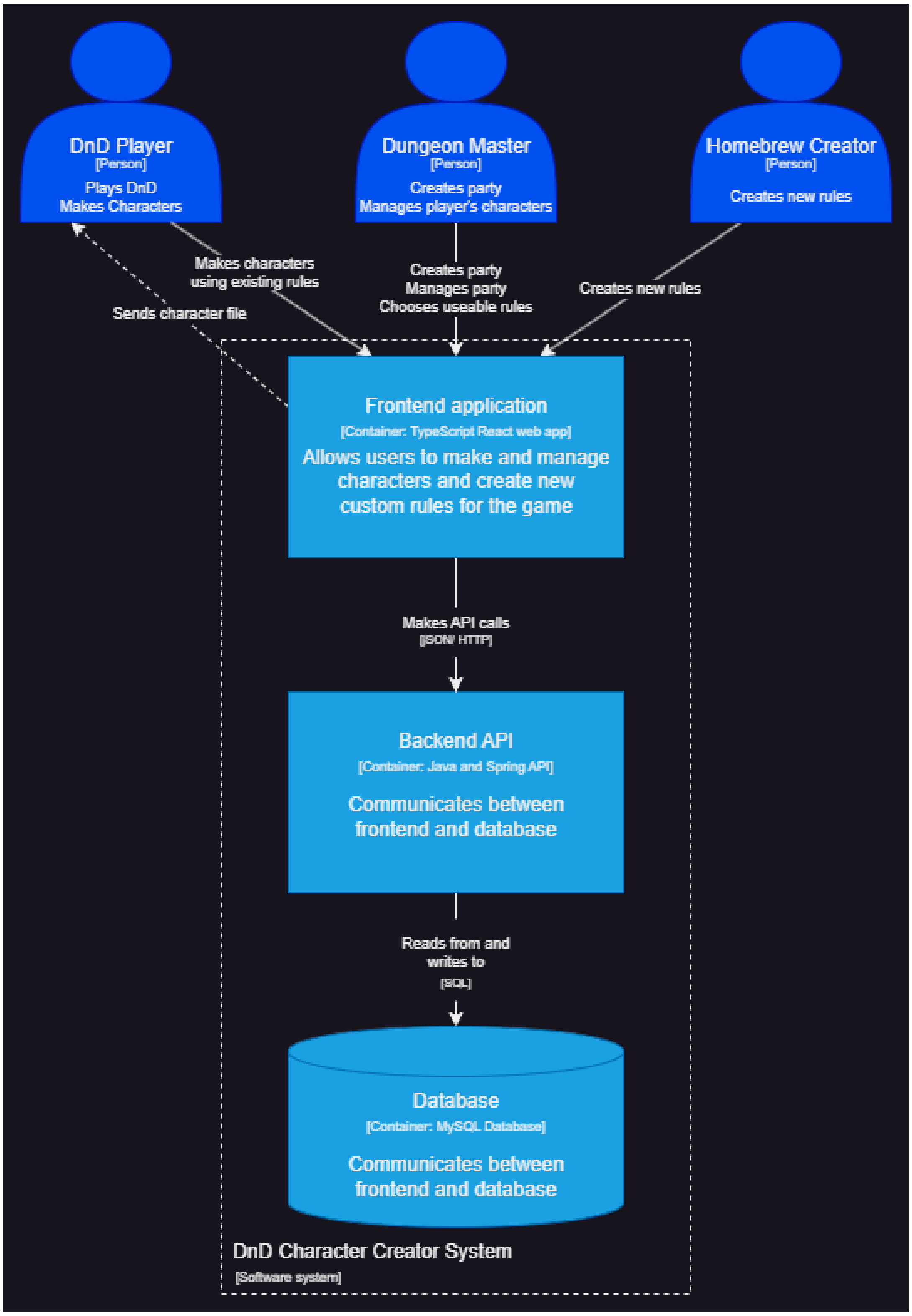
System Context Diagram



This diagram shows a very basic overview of my project. There are three types of user:

1. DnD Player: The player uses the system to create their character. This can be done within the context of a party, created by a Dungeon Master, or independently. When a player is done creating their character, the system will send them a downloadable file that contains all the data for their character.
2. Dungeon Master: The DM uses the system to create a group, or party, that the players can join. The DM can then select which races and classes (from both official and homebrew content) the players can and can't select when creating a character. They can also see what characters the players have created and approve or deny them.
3. Homebrew Creator: The Homebrew Creator uses the system to create custom (homebrew) races and classes that other people can use if they want to.

Container Diagram



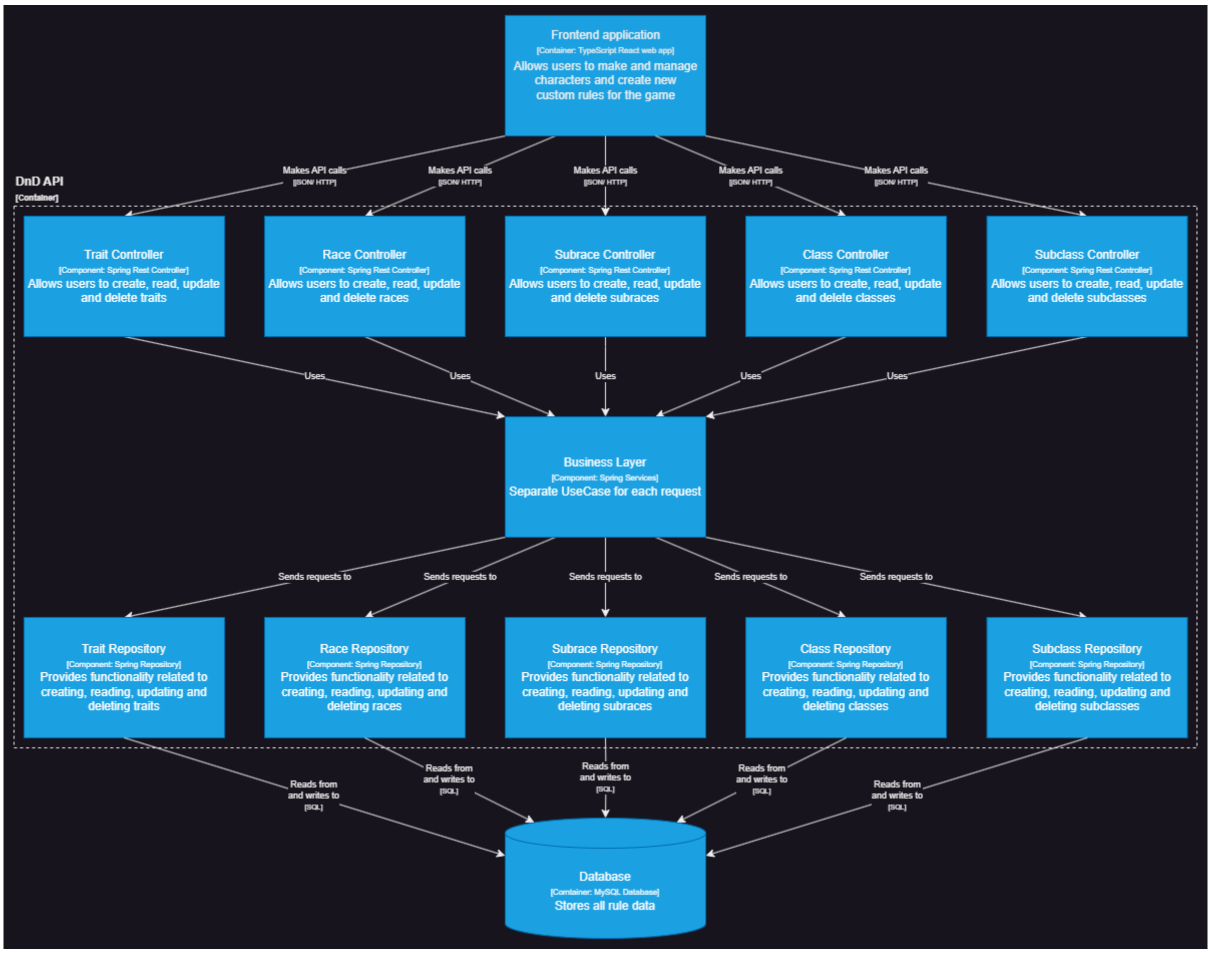
This diagram shows the D&D Character Creator System in more detail.

The frontend application is the website the users use to interact with the system. This component is a TypeScript React web app.

The backend API is the API the website uses to get data from the database and find out which races belong to a subrace for example. This Component is a Java and Spring API.

The Database is where the API can get data like rules from, and how the API can store data. The database stores all the official rules and any homebrew rules users create. This component is a MySQL database.

Component diagram


Frontend application
[Container: TypeScript React web app]
Allows users to make and manage characters and create new custom rules for the game
Makes API calls [JSON/ HTTP]
DnD API
[Container]
Trait Controller
[Component: Spring Rest Controller]
Allows users to create, read, update and delete traits
Race Controller
[Component: Spring Rest Controller]
Allows users to create, read, update and delete races
Subrace Controller
[Component: Spring Rest Controller]
Allows users to create, read, update and delete subraces
Class Controller
[Component: Spring Rest Controller]
Allows users to create, read, update and delete classes
Subclass Controller
[Component: Spring Rest Controller]
Allows users to create, read, update and delete subclasses
Uses
Business Layer
[Component: Spring Services]
Separate UseCase for each request
Sends requests to
Trait Repository
[Component: Spring Repository]
Provides functionality related to creating, reading, updating and deleting traits
Race Repository
[Component: Spring Repository]
Provides functionality related to creating, reading, updating and deleting races
Subrace Repository
[Component: Spring Repository]
Provides functionality related to creating, reading, updating and deleting subraces
Class Repository
[Component: Spring Repository]
Provides functionality related to creating, reading, updating and deleting classes
Subclass Repository
[Component: Spring Repository]
Provides functionality related to creating, reading, updating and deleting subclasses
Reads from and writes to [SQL]
Database
[Container: MySQL Database]
Stores all rule data

This diagram shows the Backend API in more detail. It shows the different controllers the backend has and what their purpose is.

Every controller is in charge of a single part of a character. The traits are the specific abilities a race, class, subrace, or subclass can have.

The business layer is shown as a single component in this diagram for simplicity. In reality, every controller has a separate UseCase it can call upon for each request it receives. For example, the Trait Controller has a CreateTraitUseCase, a GetAllTraitsUseCase, a GetTraitUseCase and a DeleteTraitUseCase. If I were to show all of these UseCases separately, the diagram would become very cluttered and hard to read, hence why I made the decision to simplify that part of the application in the diagram.

The persistence layer has a repository for each controller. Each repository manages their own database table.