**ORM Research**

# Introduction

The aim of this document is to settle which ORM would be suited best for the Atos Group Project [re.shape] is conducting during Semester 6 at Fontys ICT.

The MVP of the project is to build a working prototype of the Business Card Game Atos has internally developed, so that the process of playing the game becomes seamless and fully available online.

Contents

[Introduction 1](#_Toc96614318)

[1. Requirements 2](#_Toc96614319)

[2. Possible ORM Choices 2](#_Toc96614320)

[3. Comparison 3](#_Toc96614321)

[4. Conclusion 3](#_Toc96614322)

# Requirements

It is firstly important to understand what an ORM is and why is it useful. An ORM refers to ‘Object-relational mapping’ and is a technique used for converting data between incompatible type systems using object-oriented programming languages. Moreover, ORMs are mainly used in order to avoid writing SQL statements, as they provide functions for the operations, making the whole process faster.

Because we have chosen to base our back end on Node.JS and NestJS as a framework, we will be looking for ORMs that fit those two. It is, of course, important to look for a few specific requirements when searching for such a component:

* Popularity – a popular ORM can mean that lots of developers are using it, therefore, the common issues that may appear have already been solved
* Activity/ Development on the tool – updates and changes to the tool mean that the developers are still active, adapting the tool itself to the market changes and making it something not only for the present, but also for the future
* Documentation – documentation is very important in such a case, as depending on our experience with such a tool we will need to know how it functions and how to properly use it
* Stability – this is a very important factor as we do not want any bugs or issues within the application itself caused by any of the frameworks and tools that we are using

# Possible ORM Choices

Considering everything that has been stated at point 1 of the document, we will be selecting a number of possible ORMs that could be used for this project, we will be doing a comparison and then make the best choice.

* Prisma
* TypeORM
* MikroORM
* Sequelize

# Comparison

This chapter of the document will show a comparison between the benefits of each tool mentioned earlier, at point 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | Prisma | Sequelize | MikroORM | TypeORM |
| Popularity | High | Very High | Medium-High | Very High |
| Stability | Good (newer tool) | High (older on the market) | Good (new on the market) | High (older on the market) |
| Active Development | High Activity | High Activity | Very High | High (lately Low) |
| Documentation | Good (for Node.JS & NestJS) | Good (for Node.js & NestJS) | Good (for Node.JS & NestJS) | Good (for Node.js & NestJS) |
| Supported DBs | MySQL, PostgreSQL, MSSQL, & SQLite (MongoDB, CockroachDB | MySQL, MariaDB, SQLite and Microsoft SQL Server | MySQL, MariaDB, PostgreSQL and SQLite | MySQL, MariaDB, PostgreSQL, CockroachDB, SQLite, Microsoft SQL Server, Oracle, SAP Hana, sql.js |
| Extra Features | * Autocompletion support * Work directly with JavaScript objects and not classes and instances * Type-safe database queries | * Familiar ORM interface and ActiveRecord usage patterns * Detailed control over transactions and how they are executed * Synchronizing database based on defined Models | * @Filters (scopes) * Bulk subscribers * Handling transactions automatically. | * Supports both Data Mapper and Active Record ORM patterns * Powerful and flexible query builder * Automatic migrations generation |

# Conclusion

As a conclusion of our research, the ORM of choice for this project will be MikroORM. As this project is going to use PostgreSQL, we could only choose between the three of our choices, Prisma, MikroORM and TypeORM. Whilst Prisma is a new ORM on the market, as well as the MikroORM, it does not provide that many important features for us to be interested in. MikroORM is therefore better in this case, even if TypeORM is older on the market and is more popular, the lack of development on it and updates lately does not supply this project with too much trust for the future, meaning that an outdated component of the project may cause different types of issues, such as security or functionality ones.