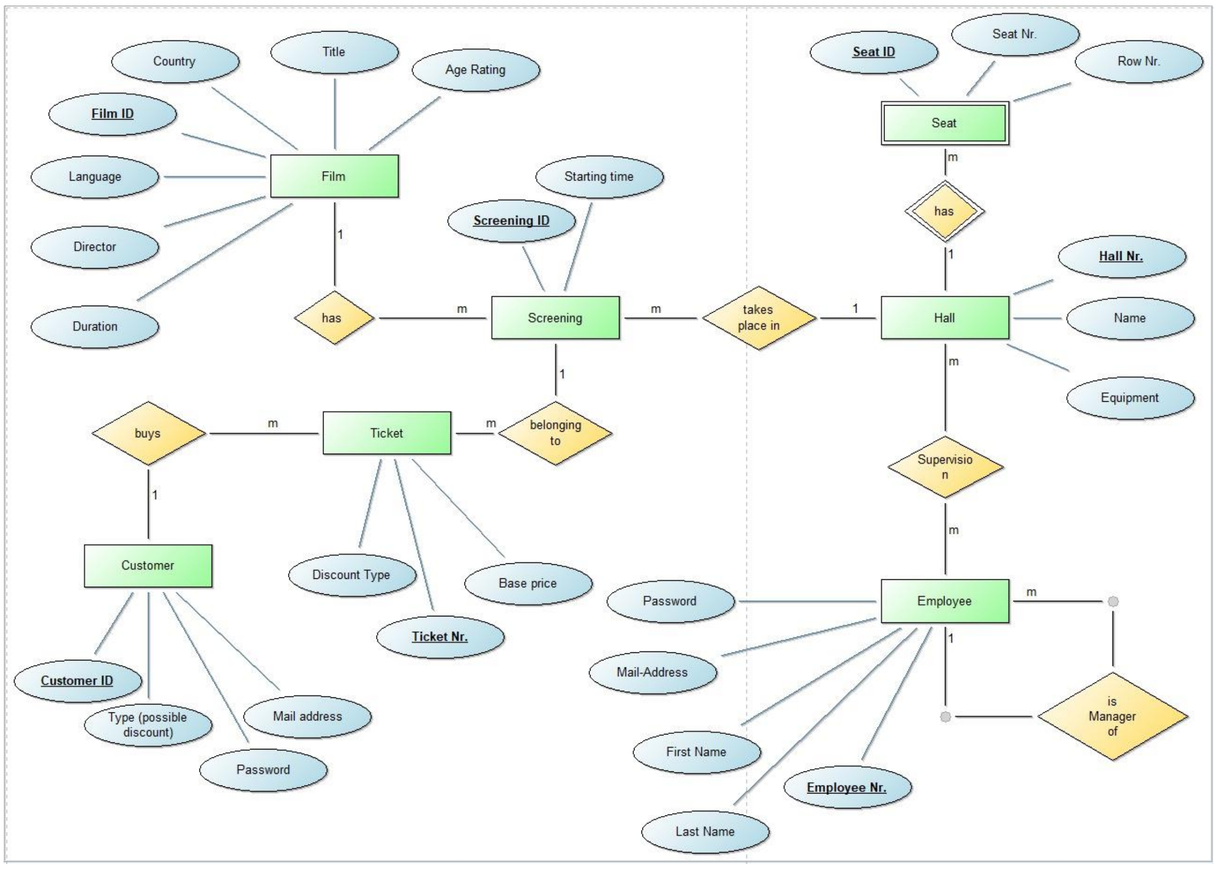
MS3 Documentation (Group 11)

1. NoSQL Database design

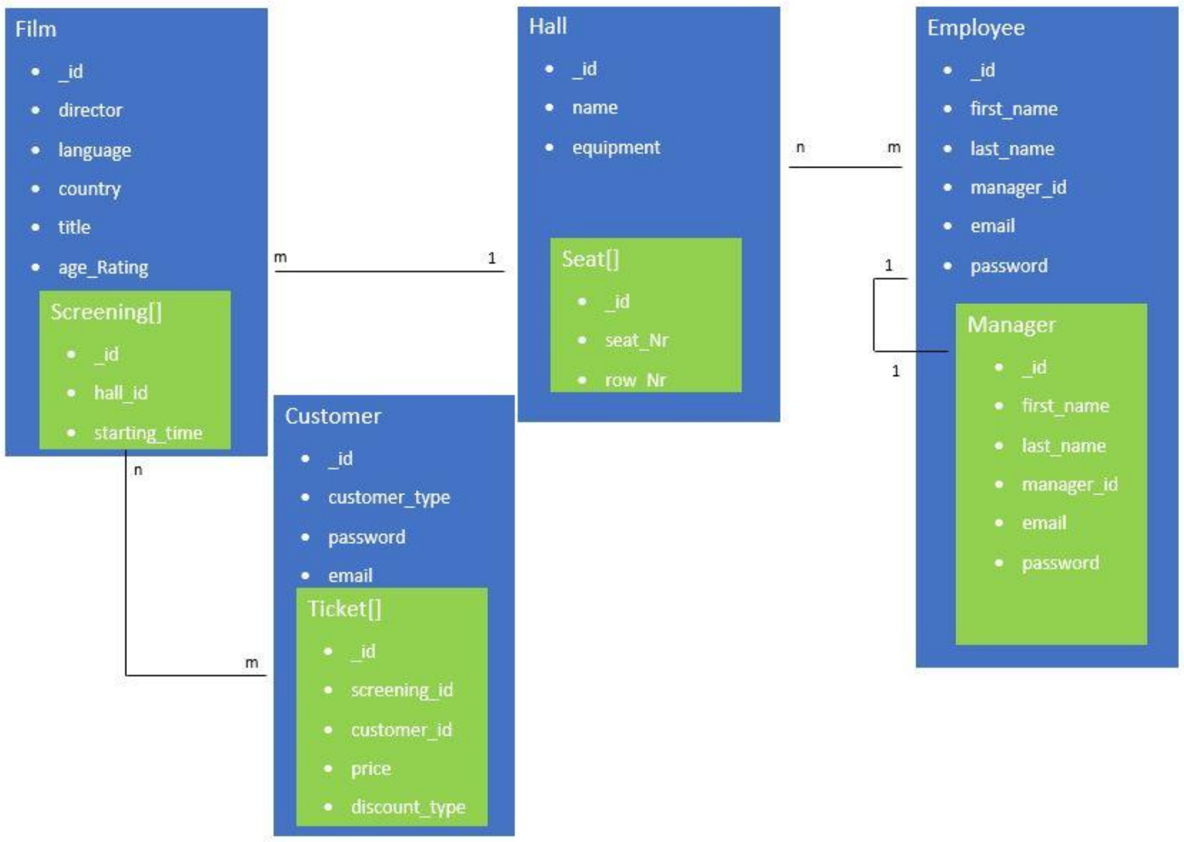
1.1. Migration of relational design to NoSQL

For Milestone 3, the task was to migrate the (My-)SQL schema to a NoSQL DBMS. Our system of choice was MongoDB, due to its unrivalled popularity and widespread support in the Java and PHP programming language. Since these are exactly the technologies we used in our project, the choice was fairly obvious to us.

However, using MongoDB involved the challenge of adapting our data model to the technology.



Clearly, this hierarchy would not work going forward - splitting up the tables as above would result in significant loss of information (consider Tickets - how would we know who they belong to?). Hence, we came up with a new, non-relational model as follows:



This design was superior to the one in Milestone 2 in regards to the lower amount of tables. However, access to the nested documents PHP would prove to be a major obstacle. As always, no change to a complex system comes without its up- and downsides.

1.2. Motivation of single steps

Concerning the tables which we deemed unviable / lacking in information on their own, we decided to integrate them in other existing tables, taking into account logical coherence. An alternative would have been to extend the fields within the documents to hold enough information on their own - we decided against this approach as we were aware that this might provide an undesirable level of redundancy within the database. Let's look at the specific changes in detail:

- **Screenings**: We integrated screenings as an array field into screenings, since according to us, this was the most straightforward implementation of the 1:n relationship they hold. Each screening holds the corresponding ID of the hall in which it is taking place.
- **Tickets**: The same goes for tickets. We chose to not explicitly model the m:n relationship between screenings and tickets, since our implemented use cases are fully functional without a dedicated reference to one or the other.
- **Seats**: Seats are modeled in the same way as screenings and tickets. A hall holds 0-n seats, with no external references or usages being required.
- **Employee** management: The 1:n self-relationship is modeled as a manager array field within employee. Each employee now holds 0-n manager IDs in this field, which in turn are simple employee IDs.

1.3. Realization of NoSQL DBMS

On top of the Technology Stack we already described in our documentation of MS2, there are now multiple additions to it.

Migration: The Java Program connects to our old database via JDBC, as in MS2. For the migration, it opens an additional connection to our new MongoDB database and inserts data it gets from the old MySQL database into the new one. This is done via the MongoDB Driver for Java.

PHP: The connection to the database in our homepage now works via the newest version of the MongoDB Driver for PHP instead of the “MySQLi”-Connector we used in MS2.

Link to the MongoDB Java Driver https://mongodb.github.io/mongo-java-driver/

Link to the MongoDB PHP Driver: https://pecl.php.net/package/mongodb

2. Data Migration

2.1. Overview

For the migration of our database, we used our Java tool in continuation of the work we already presented in Milestone 2. We created an extra application, which implements the following functions:

- **Complete migration from a MySQL schema**: Transfers the complete set of data from MySQL to our MongoDB database. To accomplish this, we created a parser (SQLMigrator.java), which selects the tuples in each table and then rearranges the syntax so they can be inserted into MongoDB sequentially.
- **Manual data entry into the MongoDB database and relevant collections**: As in Milestone 2, the manual data entry window allows the user to insert data into the desired collections. However, due to the restructuring of data mentioned in Chapter 1, we were forced to adapt how the insertion is actually handled. Nonetheless, this is completely hidden from the user and the insertion screen looks identical to the one presented in Milestone 2.
- **Dropping the MongoDB database**: In case of the migration being incomplete or another error. More granular deletion can be done via the Web interface in Employee/Admin mode.
- **Querying information regarding the database/collections**: For Milestone 3, we also provided the ability to view specific sets of information regarding the database, such as the database name, size as well as the number of collections and data sets within. This screen provided us with a simple opportunity to verify the correctness of our CRUD operations without having to use the mongo CLI or MongoDB Compass.

2.2. Initialization of the program

To run our Java application for MS3, simply download and run *executables -> MS3 -> Kino_JDBC.jar.* If you wish to compile the source code yourself, the main class can be found in *MS3-> src -> ms3kinoUI -> StartScreen.java*.

3. Implementation IS (NoSQL)

For specifics on the team members responsibilities and the use case implementation, please refer to *work_distribution -> work_distribution.docx.* For a list of our individual contributions, please refer to *work_distribution -> [surname]_work_protocol.xlsx.*