Current state:

Available user stories show that we will require a database system to handle user credentials and associate ownership of cars/parking lots to them. Further down the road more items will become visible, and require a new database schema. This is a simple issue of persistent storage, no indexing needs or big data processes are expected.

Decision tree:

small scope, all solutions offer fast retrieval/updates.

relation between entities:

- File storage requires a hand made solution which should confer to ACID to be viable as database for any serious endevour. Coding this for a temporary prototype is not a simple task and takes at least as many story points as for SQL/NoSQL solutions
- Relational databases offer alot of relational functionality and complex indexing, which is simply overkill for this task as it is.
- NoSQL key value pairs offer a decent level of relations without additional complexity.

Implementation ease:

- File storage is extremly simple technology and could be used at least temporarily, However any type of relational operations can prove painful and devour story points.
- Relational databases: requires some experience. The team has several proficient members. Java and Javascript have good adapters that can be used. Can be run as standalone or on dev/prod machines
- NoSQL databases: lightweight, easy to implement, simple key/value handling. Java and Javascript have good adapters that can be used.

Proposal:

1. Use NoSQL if teammembers have experience with it or want to try something new. Mongoose.js or MongoDB java Driver for MongoDB are well documented solution candidates.

2. For a reliable solution that can handle long term complexity a relational database can be leveraged. Sqlite3 is lightweight, and has adapters for JS(sqlite3 npm package) and Java(JDBC).

3. Try to avoid file storage solutions, they are likely to be replaced in further iterations and do not save us any story points.