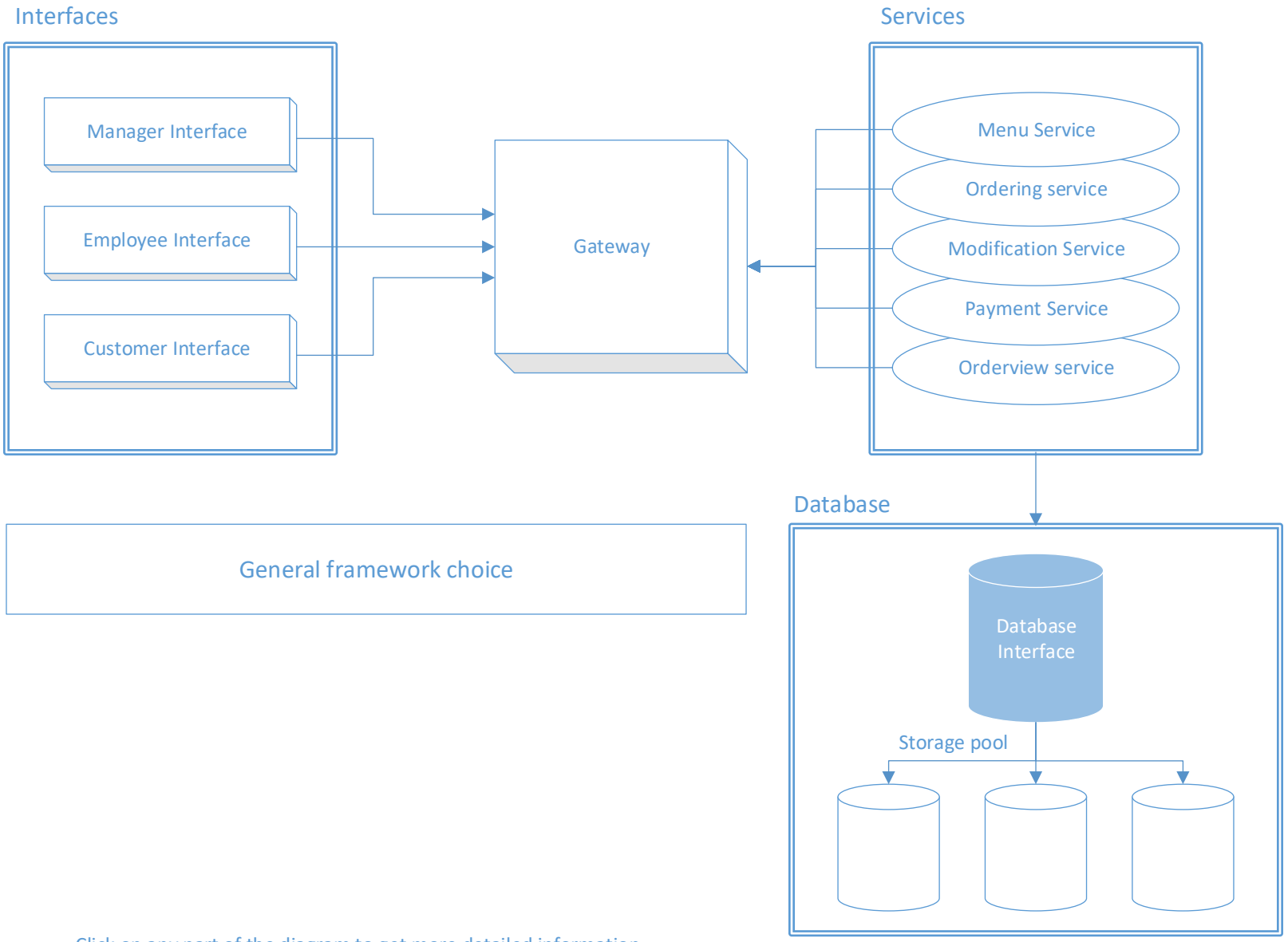
Interactive Documentation
Faros – Online Cash Register Project

Created by

Lennart Cockx
Guangming Luo

Go to Overview

Faros Online Cash Register – Technology chart



Sources

Click on any part of the diagram to get more detailed information

Framework choices

Most web frameworks are using a Model-View-Controller pattern. This pattern seperates data, business rules and the user interface. This promotes the modularization and re-use of code.

MVC frameworks itself can be divided into 2 subtypes. There are Push-based and pull-based systems. (More commonly known as Action-based and Component-based)

Since we are going to focus on Java technologies, we will limit ourselves to frameworks based on Java. Commonly used Java web frameworks are **Spring**, **JavaServer Faces** and **Vaadin**.

We do not have any experience with any of these frameworks. This means that there is a big possibility that the choice we make is not the optimal framework for our project. We believe that **good documentation** and **accessibility** will be the deciding factors in determining the best framework for us to use.

Currently, Spring MVC is the most used Java web framework, with almost 40% of the developers using this system. Various resources online also use Spring as their example framework in online courses and tutorials. Additionally, Spring has a large amount of documentation and guides available for kickstarting development and supports the basic building blocks that we need for our application.

Based on these properties, we will start development on this project using the Spring MVC framework.

Back to Overview

Customer Interface

Employee Interface

Manager Interface

Interfaces

For managers, we will use a web interface, for customers and employees, we will use both web and phone interfaces.

- Customers can either choose go to the website or download our local phone application (android/iOS). In this case, multi-platform support is needed. Apache Cordova and PhoneGap are the choices.
- For employees, Local phone application is better and convenient to use, so website app will not be designed.
- The home page and menu page for customer and employee will be different, so they will not share the same application.

Back to Overview

Customer Interface

Employee Interface

Manager Interface

User Interface

For Customer interface, we can use either Hybrid application or web application and native application. We will choose Hybrid application for better performance and convenient designing (Hybrid application fit both web page and local application).

Multi-platform support: Cordova/PhoneGap/Ionic

- Cordova also referred as Apache Cordova(formerly PhoneGap) , is open source JavaScript framework which helps you to build mobile apps with capability to access the device hardware. However, you need HTML5, CSS3, JavaScript, JQueryMobile, Sencha to build the UI. Cordova cannot be used to build UI of a mobile app. It complements other web technologies which are used to build mobile apps.
- PhoneGap is propitiatory version of Cordova maintained by Adobe. It just provides some more extra add-ons on top of existing Cordova. App built on PhoneGap can access native functions from the devices as well as the mobile operating system by using JavaScript.
- Ionic is combination of AngularJS and Cordova. Ionic is a complete open-source SDK for hybrid mobile app development. You can think of it as a full stack framework for building cross-platform mobile apps. Ionic provides all the functionality that can be found in native mobile development SDKs. Users can build their apps, customize them for Android or iOS, and deploy through Cordova.

Link to technical documentation

Back to Overview

Customer Interface

Employee Interface

Manager Interface

Employee Interface

Since local phone application is used here, we can also choose either Hybrid application or native application. For dynamic changing purpose (e.g. in menu overview page), better performance and Faster development for different platforms, Hybrid application is preferred here.

Below, you can have an overview of the advantages for Hybrid application compared to others.

	Web App	Hybrid App	Native App
Development costs	Low	Middle	High
Maintenance Updates	Simple	Simple	Complicate
User Experiences	Poor	Middle	Superior
Store or market	No	Yes	Yes
Installation	Not required	Require	Require
Cross-platform	Superior	Superior	Poor

Back to Overview

Customer Interface

Employee Interface

Manager Interface

Manager Interface

For the manage interface, its preferable to use web application to access their account and manage their restaurants or activities.

However, in Mobile Internet era, phone application is always a nice-to-have feature. Because its not necessary here, we will decide whether to make a phone application latter.

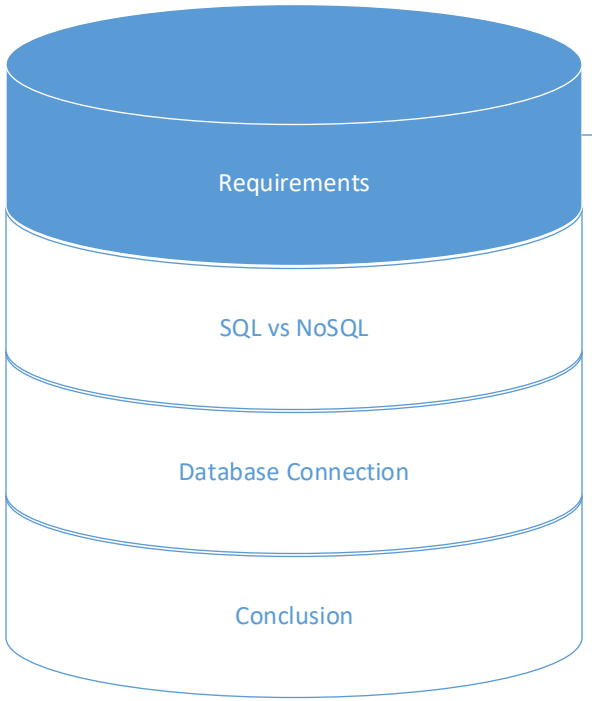
Back to Overview

Work in progress

Back to Overview

Work in progress

Back to Overview

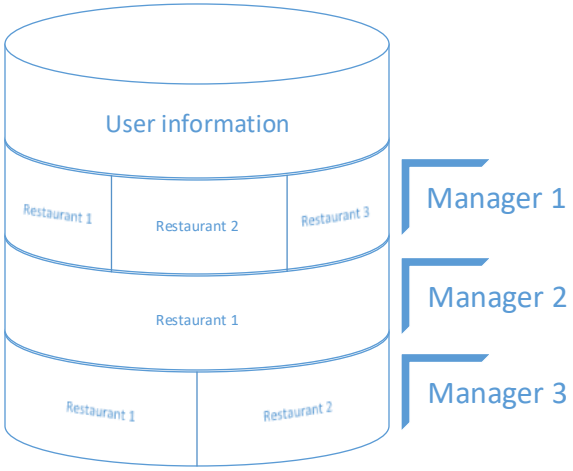


Requirements

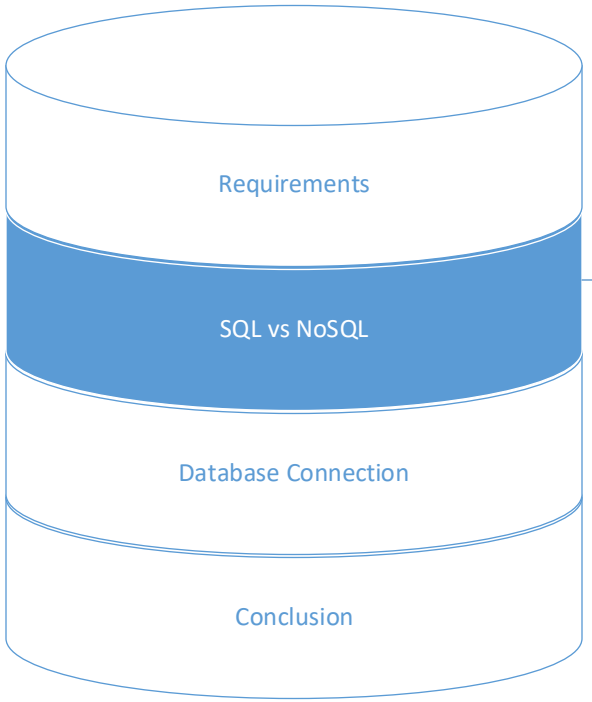
To operate, our web application needs a database. From this database we can retrieve and store information for the customers, employees and the manager.

We don’t know in advance how many managers will make use of this application. Each manager can also create and maintain a variable number of restaurants/events. This means that our database technology should be **Scalable** and **Flexible**.

Each time a user accesses the web application, their interface will dynamically request several distinct resources from the database. This process should be **Straightforward** and **Performant**.



Back to Overview



SQL vs NoSQL

There are two main types of databases, relational and non-relational.
Our research is focused on **MySQL** (relational-database) and **MongoDB** (document-based database). These two programs are commonly used in their respective categories.

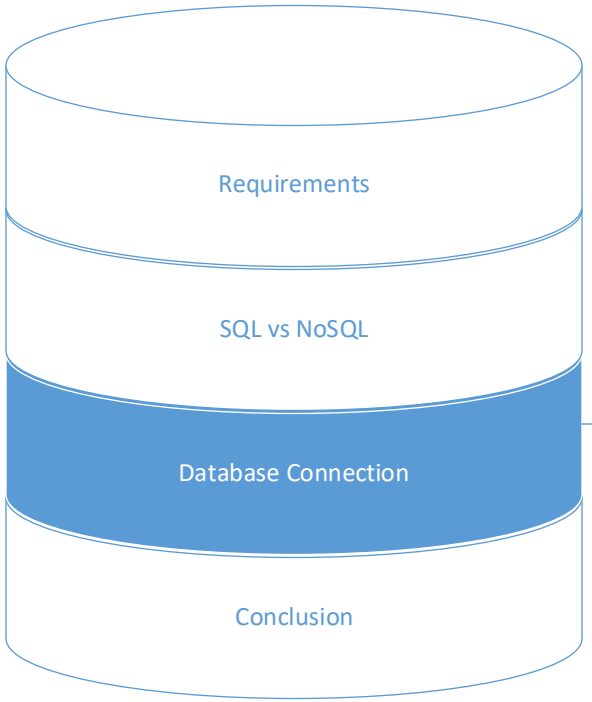
Both these options are available for **free** and are accessible through **java-based** technologies.

Flexibility
SQL records need to conform to pre-defined schemas. The format of each table is decided before data entry.
NoSQL schemas are dynamic, the format of data can be modified on the fly.

Scalability
SQL scales vertically. Storing more data will require a larger server.
NoSQL scales horizontally. This means that data is distributed over multiple servers. Using multiple smaller servers is typically less expensive than using one large server.

Link to technical documentation

Back to Overview



How to connect to the database

For connecting to our chosen database we will make use of a database connection framework. This allows us to quickly and efficiently connect to the database and retrieve and store information on it.

Hibernate and **JDBC** are two popular examples of frameworks that simplify interaction with databases. However, both of these technologies are built for use with relational databases. If we want to make use of a NoSQL system, we can use **Hibernate OGM**, which is rebuild to work with NoSQL, or search for other alternatives.

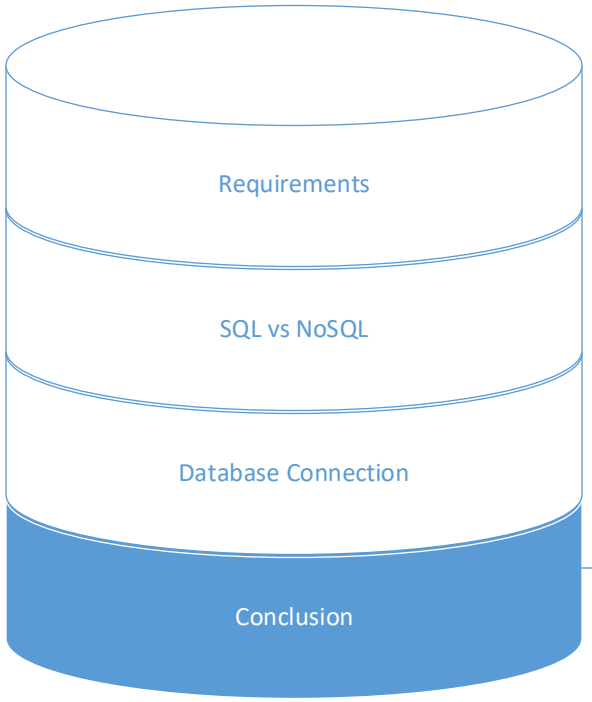
Luckily, the spring framework contains a built-in solution for communicating with NoSQL databases, namely **Spring Data**.

Spring Data has several main modules that we can make use of.

- **Spring Data MongoDB** supports communication with MongoDB databases, as the name implies.
- **Spring Data REST** makes it easier to create REST web services on top of Spring Data Repositories

Link to technical documentation

Back to Overview



Conclusion

Because of its flexibility and room for scaling MongoDB will be our preferred choice of database. However, depending on how we design the application a solution that works with an SQL database would also be possible.

For connecting this database with the framework we will use spring data, this slice of the spring framework allows the manipulation of data with NoSQL databases like MongoDB.

Back to Overview

Pattern: API Gateway (Chris Richardson, 2014) Microservices.io

When to use NoSQL vs SQL (Mimi Gentz, 2016) Azure.microsoft.com

DB-Engines Ranking (November 2016) Db-engines.com

SQL vs NoSQL, What you need to know (Eileen Mcnulty, July 2014) Dataconomy.com

Hibernate Vs JDBC (Dipti Phutela, November 2016) mindfiresolutions.com

Spring Documentation (November 2016) spring.io

Top 4 Java Web Frameworks Revealed (January 2015) zeroturnaround.com

Back to Overview