Software Architectures

Part 70: Project License Management

Project Objectives

Features

- Develop a simple web based license management system
- Manage users associated with customer companies
- Manage licenses and service contracts

Technologies

- Use MySQL RDBMS
- Use Java 11 and Quarkus Hibernate ORM
- Use Quakrus JAX-RS for web client interface
- Use ReactJS as client side technology

Requirements and Grading

- You must use one Gitlab account to manage your projects version control. Each team member must have a significant amount of commits. **Failure to show these commits means you fail your project, no kidding!**
- You must have Java unit tests for all non-trivial classes.
- You must be present on the dates announced for project reviews on Moodle.

Grading

- Database design (correctness, completeness): 10 points
- ORM and database related functionality (design, correctness, completeness, test coverage): 20 points
- REST API (functional completeness, correctness, test coverage): 20 points
- UI login and user management (completeness, design, usability): 10 points
- UI customer management (completeness, design, usability): 10 points
- UI contracts management (completeness, design, usability): 20 points
- Build and deploy process (installation from scratch on new computer): 10 points

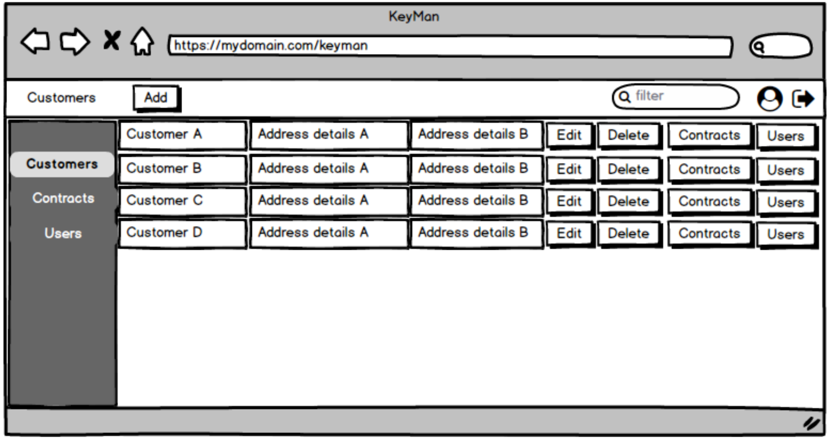
You will need at least 70 (SWB) or 50 (TIB) points to pass the lab.

Customer related Functionality

- The system shall manage customer companies
- All customer related management shall be reserved to administrators
- Administrative users may edit customer profiles, add customers, and delete them
- Customer shall have a name, department, and an address (international)
- A filter shall limit the customers shown based on the first letters of the customer name
- The list view shall show all customer attributes and have buttons to edit, delete, show all related contracts and all related users.

Customer related Functionality
Customer overview screen mock

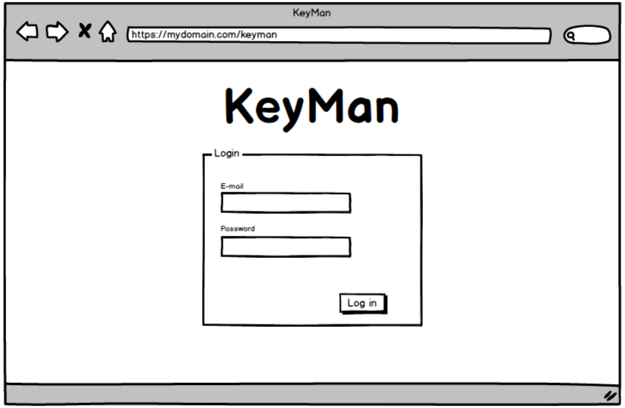
Note: This is a mock and shall give you a rough idea of the user interface. Make it better!



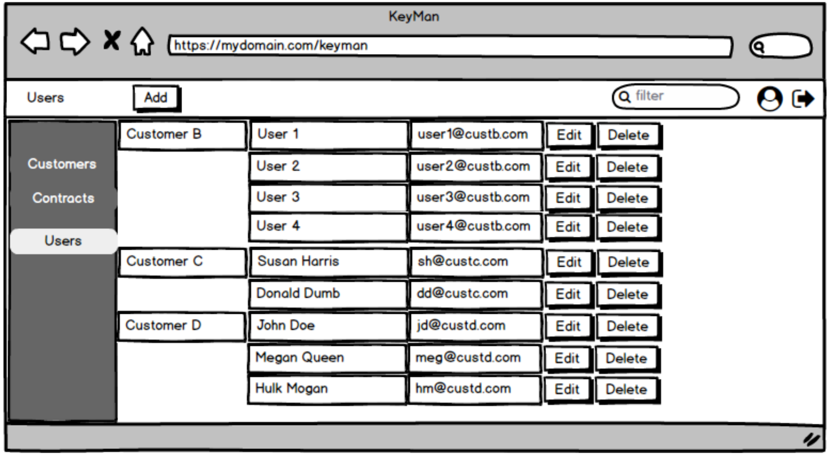
Customer A	Address details A	Address details B	Edit	Delete	Contracts	Users
Customer B	Address details A	Address details B	Edit	Delete	Contracts	Users
Customer C	Address details A	Address details B	Edit	Delete	Contracts	Users
Customer D	Address details A	Address details B	Edit	Delete	Contracts	Users

User related Functionality

- The system shall manage users related to companies; a company can have many users
- Users must be able to login, edit their profile and change their password
- Administrative users may edit other users profiles, add users, and delete them
- Users shall have a first and last name, a login name, an e-mail address, and two phone numbers
- Users may be marked as administrators

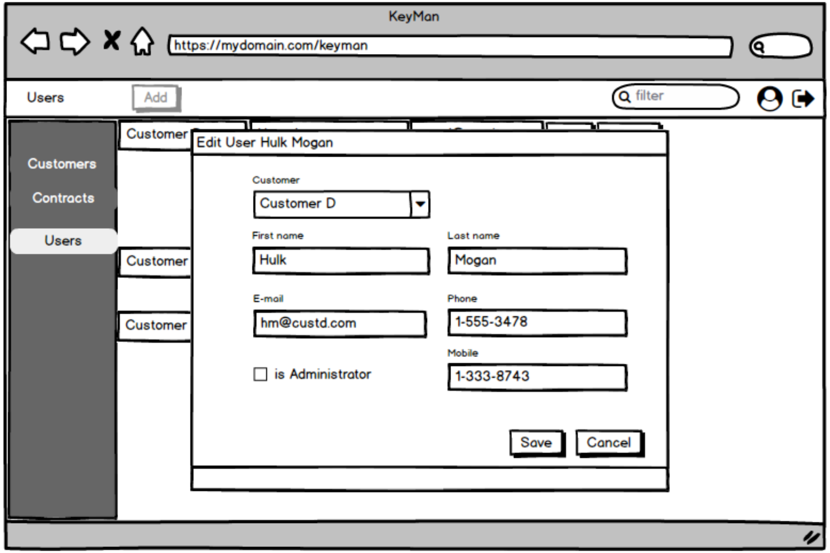


User related Functionality
User list overview screen mock



Customer B	User 1	user1@custb.com	Edit	Delete
	User 2	user2@custb.com	Edit	Delete
	User 3	user3@custb.com	Edit	Delete
	User 4	user4@custb.com	Edit	Delete
Customer C	Susan Harris	sh@custc.com	Edit	Delete
	Donald Dumb	dd@custc.com	Edit	Delete
Customer D	John Doe	jd@custd.com	Edit	Delete
	Megan Queen	meg@custd.com	Edit	Delete
	Hulk Mogan	hm@custd.com	Edit	Delete

User related Functionality
User details edit screen mock



Edit User Hulk Mogan

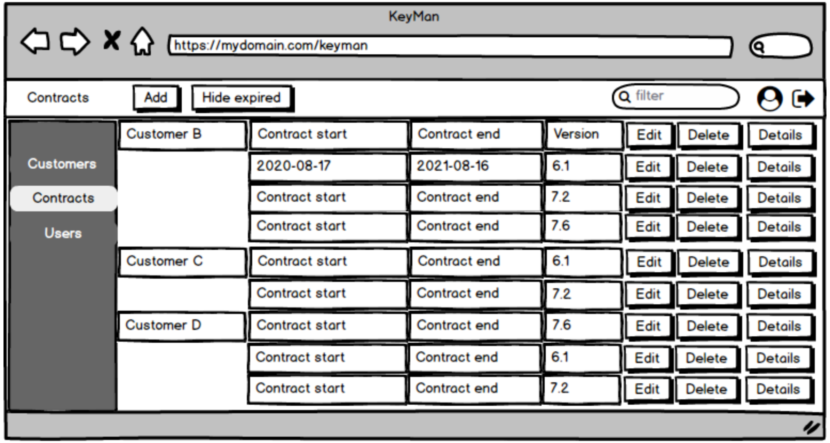
Customer: Customer D	
First name: Hulk	Last name: Mogan
E-mail: hm@custd.com	Phone: 1-555-3478
☐ is Administrator	Mobile: 1-333-8743

Service Contract related Functionality

- The system shall manage service contracts
- All service contract related management shall be reserved to system administrators
- Administrative users may add, edit and delete service contracts
- A service contract is bound to a customer; typically a customer can have many service contracts
- A service contract shall have a start date, an end date, three IP numbers, a version, three numerical feature fields, two users, and a large text field for the license key The 2nd and 3rd IP number fields are only enabled if the previous IP number has been filled in
- A service contract is associated with at most two users from the associated company
- Users associated with a service contract may see its details, but must not be able to change them except for the IP numbers and the version
- If the service contract is expired, users may just change the IP numbers (admins still may change everything)

Service Contract related Functionality

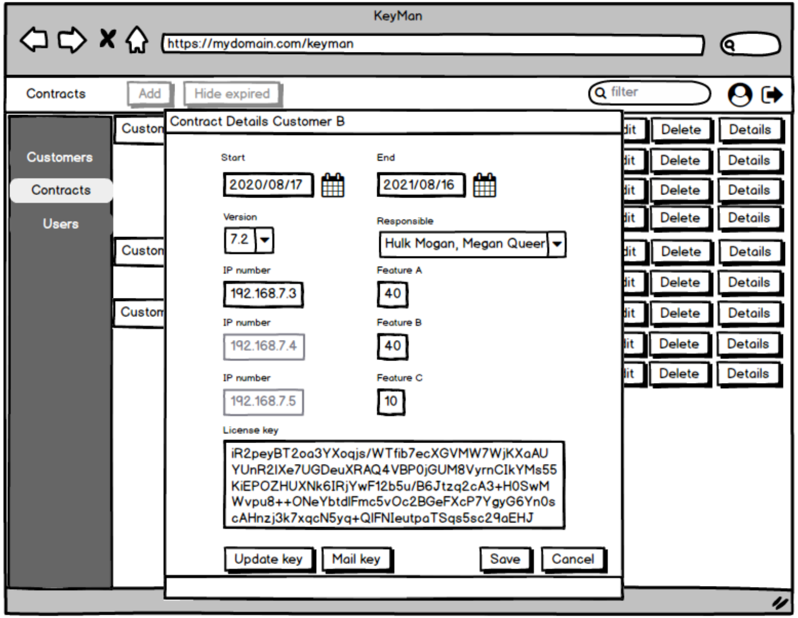
Service contract list view



Customer B	Contract start	Contract end	Version	Edit	Delete	Details
	2020-08-17	2021-08-16	6.1	Edit	Delete	Details
	Contract start	Contract end	7.2	Edit	Delete	Details
	Contract start	Contract end	7.6	Edit	Delete	Details
Customer C	Contract start	Contract end	6.1	Edit	Delete	Details
	Contract start	Contract end	7.2	Edit	Delete	Details
Customer D	Contract start	Contract end	7.6	Edit	Delete	Details
	Contract start	Contract end	6.1	Edit	Delete	Details
	Contract start	Contract end	7.2	Edit	Delete	Details

Service Contract related Functionality

Service contract detail view



Contract Details Customer B

Start: 2020/08/17
End: 2021/08/16
Version: 7.2
Responsible: Hulk Mogan, Megan Queer
IP number: 192.168.7.3
Feature A: 40
IP number: 192.168.7.4
Feature B: 40
IP number: 192.168.7.5
Feature C: 10

License key:

```
iR2peyBT2oa3YXoqjs/WTfib7ecXGVMW7WjKXaAU
YUnR2IXe7UGDeuXRAQ4VBP0jGUM8VyrnCIkYMs55
KiEPOZHUXNk6IRjYwF12b5u/B6Jtzq2cA3+H0SwM
Wvpu8++ONeYbtdlFmc5vOc2BGeFXcP7YgyG6Yn0s
cAHnzj3k7xqcN5yq+QIFNIeutpaTSqs5sc29aEHJ
```

Deployability

1. You need to keep in mind that your application later on needs to run on some server, not just localhost:8080
2. Avoid using absolute URLs in your code like https://localhost:8080/xxx/yyy, use /xxx/yyy instead
3. Make sure that the base URL is configurable when you start the server, for example via command line parameters or environment variables
4. Make sure the database user and password are configurable when you start the server, for example via command line parameters or environment variables

Outline of the development process

1. Create an UML class diagram for the entities in the database
2. Write the enitiity classes (the model) in Java
3. Using Quarkus create the database schema and fill it with some test data
4. Develop the ORM classes to access the database entities
5. Develop the resource classes for the server side REST interface
6. Develop the client side ReactJS code for the login screen
7. Develop screen by screen the rest of the client side functionality

Note: Don't forget to write and run unit tests for all non-trivial server side classes!

Note: Don't forget to regularly check in your code in GitLab! You'll fail if you do not comply!