Project Assignment

Internet Applications Design and Implementation 2022/23

Change log

- 2022-10-04: Initial version, still up for discussion and receiving feedback.
- 2022-10-06: Added submission details and registration form.
- 2022-10-11: Incorporated answers to questions from discord (tagged with **NEW:**)

Introduction

The topic for this year is an application to manage a fleet of delivery trucks in a carrier company. The core of our concerns lies in managing the entry, exit, and change of state of packages in the system as they are shipped, transported between distribution hubs, and delivered to the destination. The company has a series of storage locations (hubs) where packages are delivered and picked up.

The actors in this system are the following: Client, Driver, Hub worker, and Manager.

Each package has a history of states in the system, which is a ordered and timed list of states. The states are the following: Awaiting entry (**NEW**: it is nowhere in the network yet), in storage (with relation to a hub), in transit (with relation to a driver), delivered, returned to the source, lost or damaged.

Every time a package changes state, a new entry is added to the history of states, and messages are exchanged between the different roles in its process. Further messages can be exchanged between clients and the other users in the system. A client can ask questions about the status of a package, and a staff member (driver, hub worker, and manager) can respond to the client.

Technical Details

Server-side application

Your server-side application should implement a layered architecture to make available a REST API for the resources of the system. The resources that are visible in the API are the following: packages, messages, hubs, and trucks.

Data resources needed to make the application work do not need to be fully managed by the API. All main operations are listed below and all auxiliary data can be inserted into the database using seed data initializers of SQL scripts.

- Packages:

  The information of a package is the following: an identifier, a description (box/envelope/unusual size), dimensions (w, h, l), weight, origin, and the destination address.

The package API should follow the REST architectural style and should be implemented using the Spring framework. The API should include all operations concerning this resource to support the scenario.

- Messages:

  A message has one source contact (username or email), a destination contact (username or email), a subject, a body, and a timestamp. The API should include all operations concerning this resource to support the scenario.

  All messages that are added to the system should correspond to an email message sent to the destination contact. See Spring documentation to add an e-mail server connected to your application.

  NEW: All messages have a pointer to the previous message that allows one to build a thread of messages. A new thread is created with an invalid identifier of the previous message. Shipments have a thread of messages associated with them where the root message corresponds to the creation of the shipment.

- Hubs:

  The API should allow the listing of packages in hubs and the listing of hubs. The management of hubs is not part of the API. The hubs are identified by a code, a name, an address, and a description.

- Trucks:

  The API should allow the listing of packages in trucks and the listing of trucks. The management of trucks is not part of the API. **NEW:** Each truck has one driver and one driver only drives one truck.

Requirements for the Server Application

The application should be implemented using the Spring framework and Kotlin. The API should be documented using OpenAPI 3.0. The application should be developed in a layered architecture with a persistent database (H2 in a file or MySQL).

The application should implement security rules as prescribed in the course lectures. The security rules should follow the examples below and will be fixed at a later stage.

Requirements of the Client Interface

To be released later

Examples of security rules

- A client can create a shipment.

- A client can track the status and history of their shipments.
- A client can also cancel a shipment if it is under the status `Awaiting entry`.
- A client can only see the status of a shipment, and the history of the states of a shipment.
- A driver can only change the status of a shipment to `in transit` or `delivered`.
- A hub worker can only change the status of a shipment to `in storage`.
- A manager can change the status of a shipment to any of the possible states.
- A manager can also change the destination of a shipment.
- A manager can create a shipment (**NEW:** on behalf of any user)
- A manager can track the status of any package.
- **NEW:** No one can edit or delete messages from a mailbox

Working Teams

The project assignment should be done by teams of 4. If you cannot make a team of 4, talk to the instructor to authorize other solutions.

Deliverables

All deliverables should be submitted by pushing them to the repository of the project in github classroom.

1. Database schema
2. OpenAPI (a pdf file with the specification)
3. Server-side application (a commit in the repository)
4. **NEW:** Automatic unit and integration tests in a continuous integration style
5. Client-side application (a commit in the repository)
6. Presentation
7. Report
8. Video with Demo

Submission details

You should register your team in the following google forms. You will need to have a GitHub identifier (use the campus address to have the educational benefits).

https://forms.gle/1KkfATcL311ZR6Ns7

First submission deliverables (NEW)

You should have a commit tagged as "First" (A GIT TAG) in your repository containing the source code for the server-side application, its tests, and pdf documents with items 1 to 4 above.

Second submission deliverables (NEW)

You should have a commit tagged as "Final" (A GIT TAG) in your repository containing the source code for the server-side and client-side applications, their tests, and documents for all items above. The video can be uploaded to YouTube (unlisted) and the link can be part of the report.

Report Template

The report is a markdown document, submitted to the project repository, that should contain the following sections:

- Architecture description
- Database schema (ER Diagram)
- OpenAPI summary
- Security rules implemented
- User Stories
- IFML Diagram
- Links to the commits in the repository that represent the submission of the different parts of the project.
- Group dynamics: how the different elements of the team contributed to the project.
- Lessons learned: Highs and lows in the project development
- Auto-evaluation (also in numbers) "We think this work is worth X"

You may include figures, diagrams, and code snippets in the report.

Important Dates

First submission: 2022-11-01

Second submission: 2022-11-30

(**NEW:**) Evaluation: 2022-12-19 to 2022-12-21