COMP1610 – Programming Enterprise Components Report

Your name	Yauhen Bichel	Your Student ID	001185491

Other members in the Group:	Ivan Huliyev	Their IDs	001181952
	Yahya Chahine		001178039
	Nurkaiyr Yedige		001196862

1. BRIEF STATEMENT OF FEATURES YOU HAVE IMPLEMENTED

(THIS SECTION SHOULD BE THE SAME FOR ALL MEMBERS OF THE GROUP)

Feature	Status	Your Comments
Functionality A	Fully completed ☒ Partially completed ☐ Having bugs/Not working ☐ Not implemented ☐	Designed and implemented
Functionality B	Fully completed ☒ Partially completed ☐ Having bugs/Not working ☐ Not implemented ☐	Designed and implemented
Functionality C	Fully completed ☒ Partially completed ☐ Having bugs/Not working ☐ Not implemented ☐	Designed and implemented
Functionality D	Fully completed ☐ Partially completed ☒ Having bugs/Not working ☐ Not implemented ☐	Server side: Constraints are implemented and applied with validation of request Client side: not implemented
Functionality E	Fully completed ☒ Partially completed ☐ Having bugs/Not working ☐ Not implemented ☐	RESTful service is designed and implemented Desktop app is implemented
Functionality F	Fully completed ☒ Partially completed ☐ Having bugs/Not working ☐ Not implemented ☐	Message system and mail service are implemented. Created gmail mailbox as holiday booking system mailbox

Functionality G	Fully completed ☐ Partially completed ☒ Having bugs/Not working ☐ Not implemented ☐	Prioritization is designed and implemented.

Link to recorded video (if you record your application before submitting the report) The recorded video will demonstrate the implemented product as group. It is roughly 10 minutes. The video you upload on Panopto should follow this naming convention – "Surname1-Surname2- etc." You should include the surnames of all group members of all group members in the video name, so we can easily identify the group.
https://gre.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=00e89ecb-553f-4bcf-b795-ae83007e73b6

2. PART A (COMPLETED AS A GROUP) (10%)

a. **Design Documentation**: This section should contain the design documentation that you created as a group, which should contain an Entity Relationship Diagram outlining the database structure and an architecture diagram which shows the overall setup of the system.
b. **Screenshots:** Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.
c. **Evaluation (approx. 600 words)**: An evaluation of the evolution of your application. You should discuss any problems you had during implementation. You should be critical (both positive and negative) of your implementation. Be prepared to suggest alternatives. Discuss how your final implementation could be improved

Design Documentation

The Holiday System application contains the following functionality

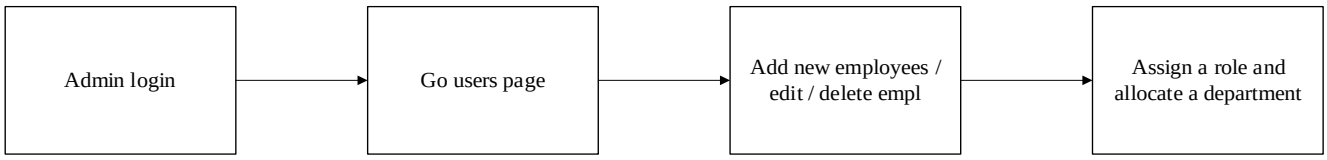


Figure 1 Functionality A: admin adds new employee

Figure 2 Functionality B: employee submit a holiday request

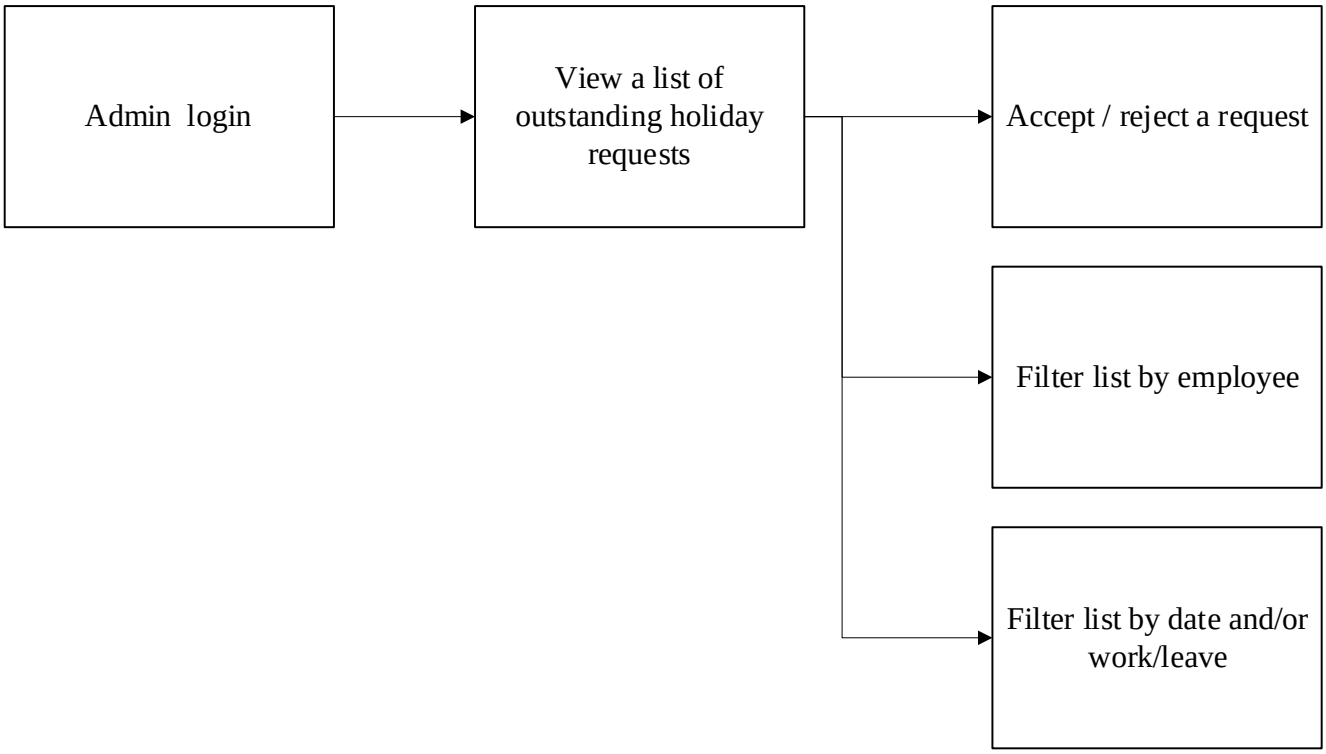


Figure 3 Functionality C: admin views requests and accept/reject them

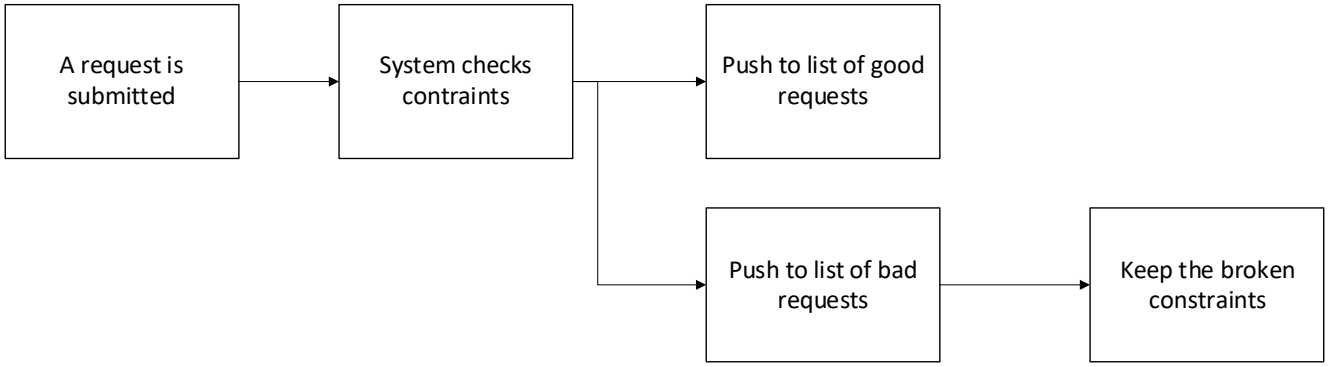


Figure 4 Functionality D: a system validates a submitted request

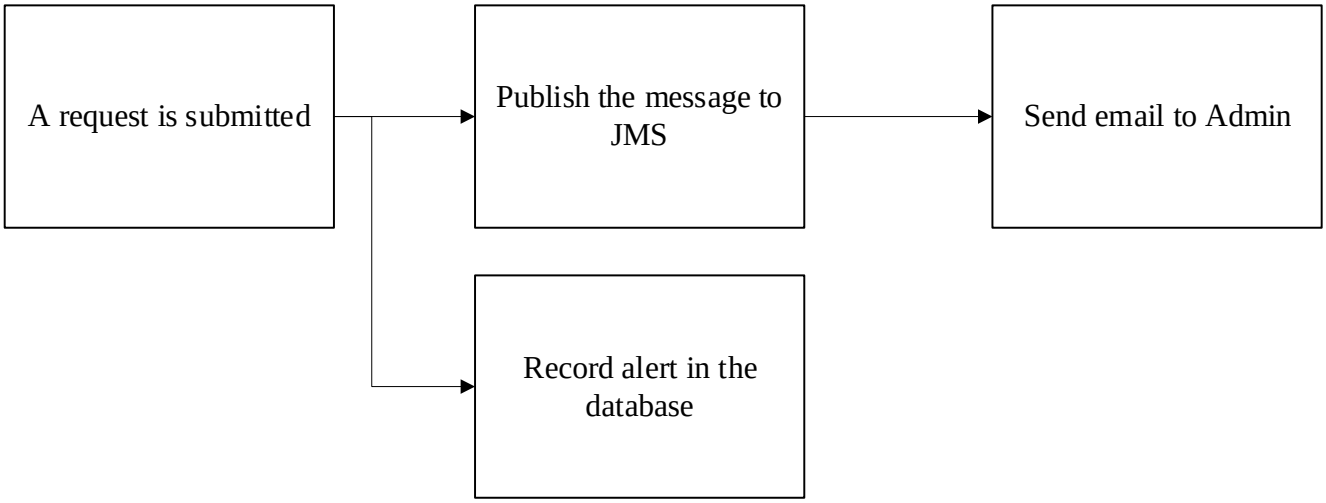


Figure 5 Functionality F: notify admin via email using JMS and database

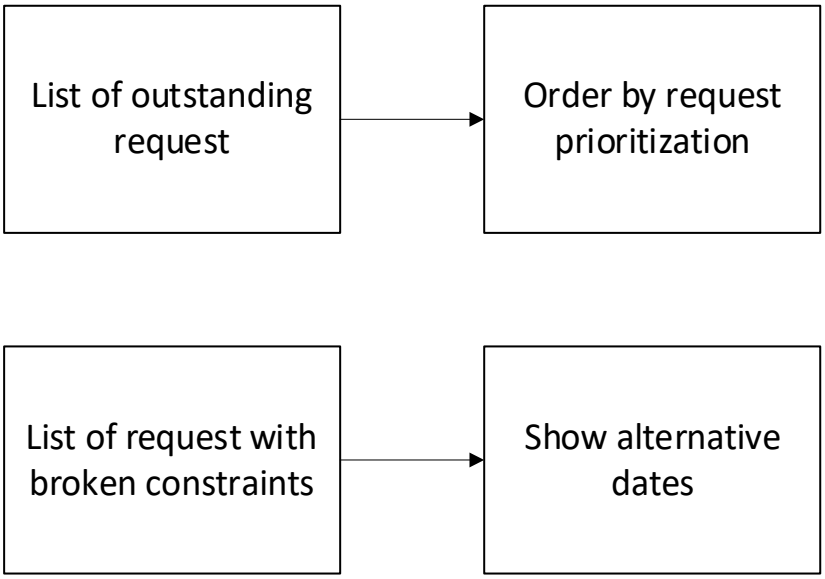


Figure 6 Functionality G: prioritization of requests and alternative dates

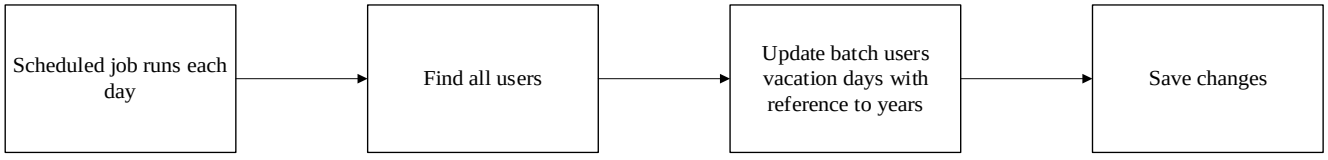


Figure 7 Scheduled job to update batch of vacation days for employees

High level network model of holiday system is demonstrated in Figure 8.

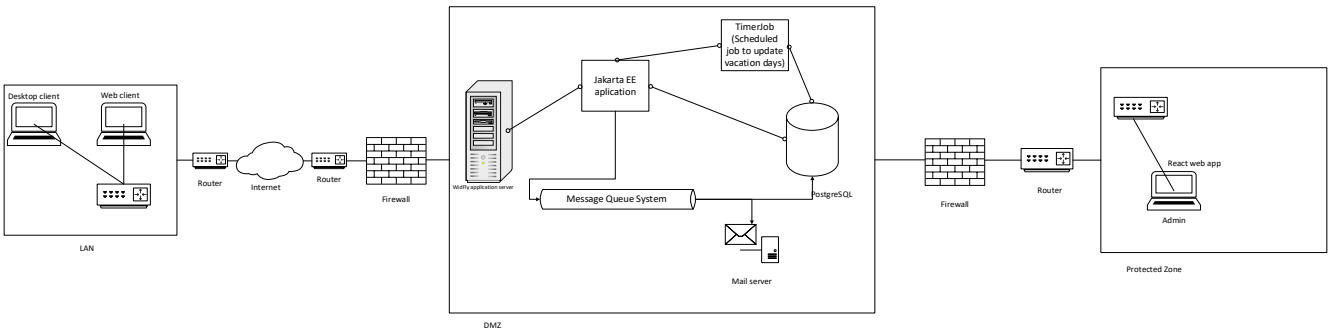


Figure 8 Simple Network Model

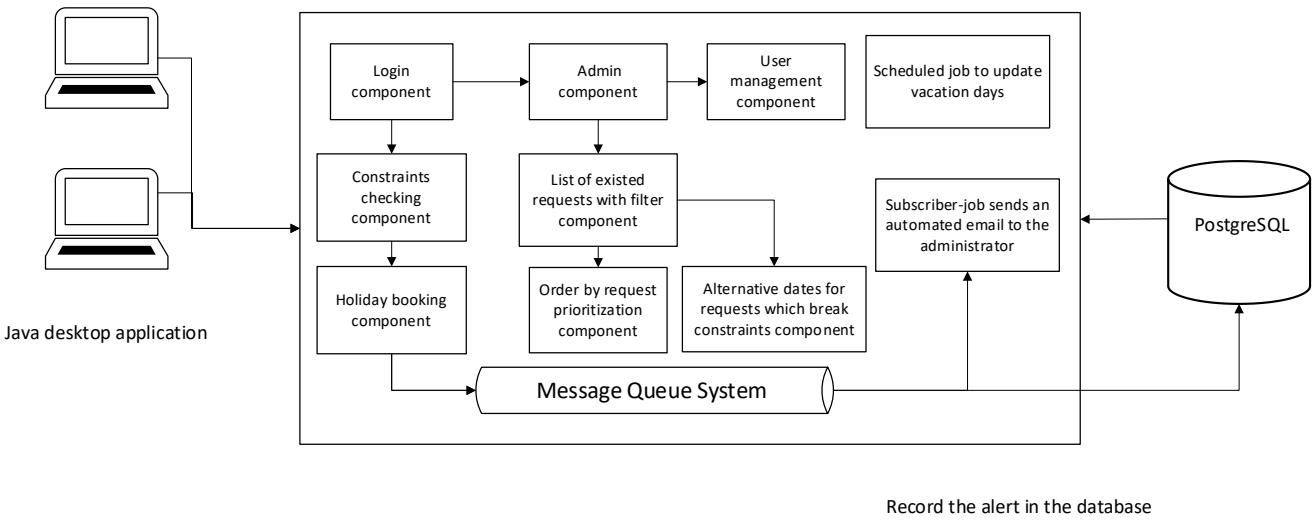


Figure 9 System Design

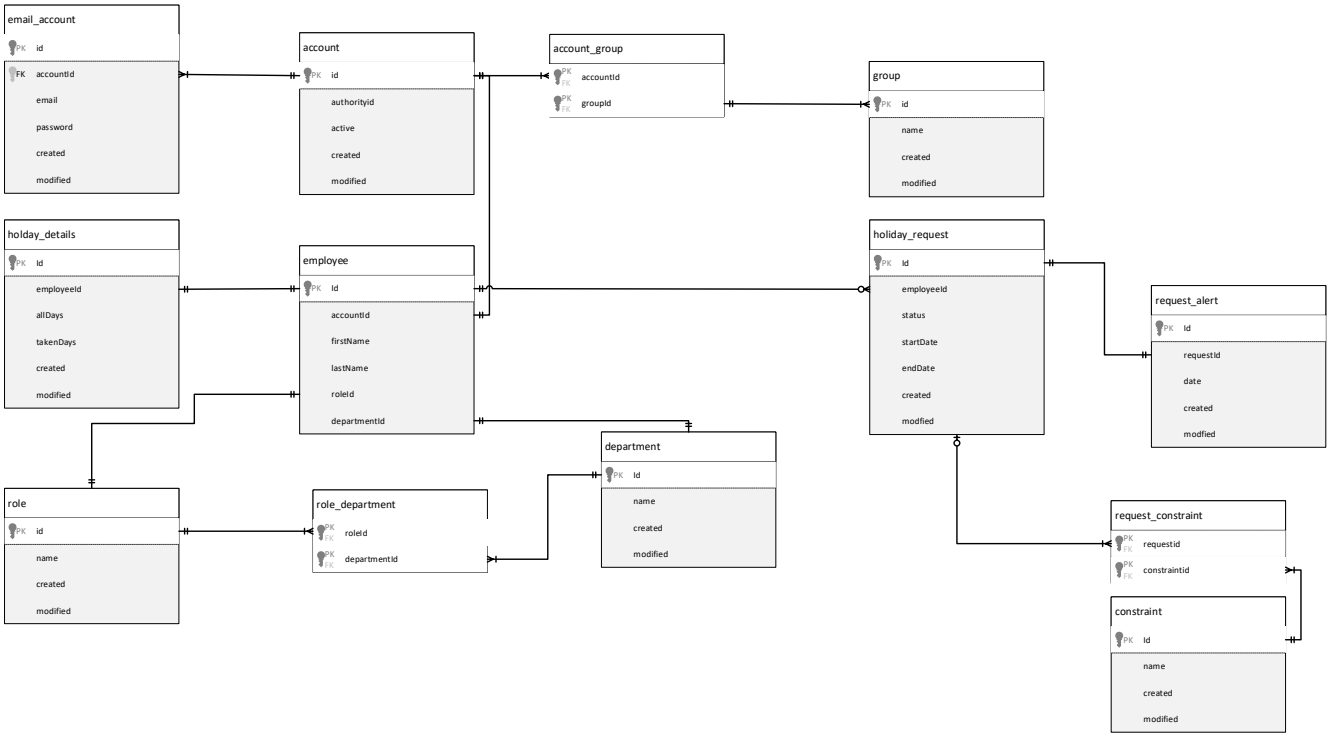


Figure 10 Entity Relationship Diagram

Screenshots

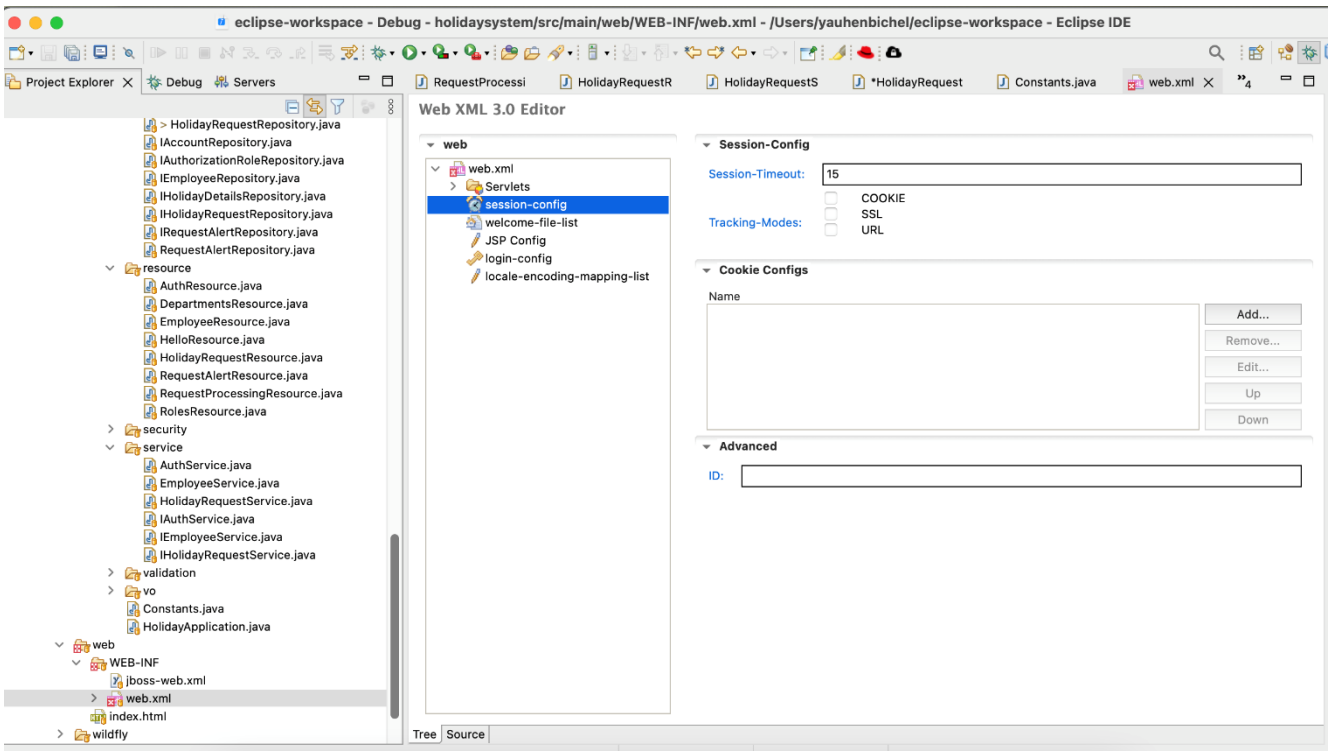


Figure 11 Session timeout

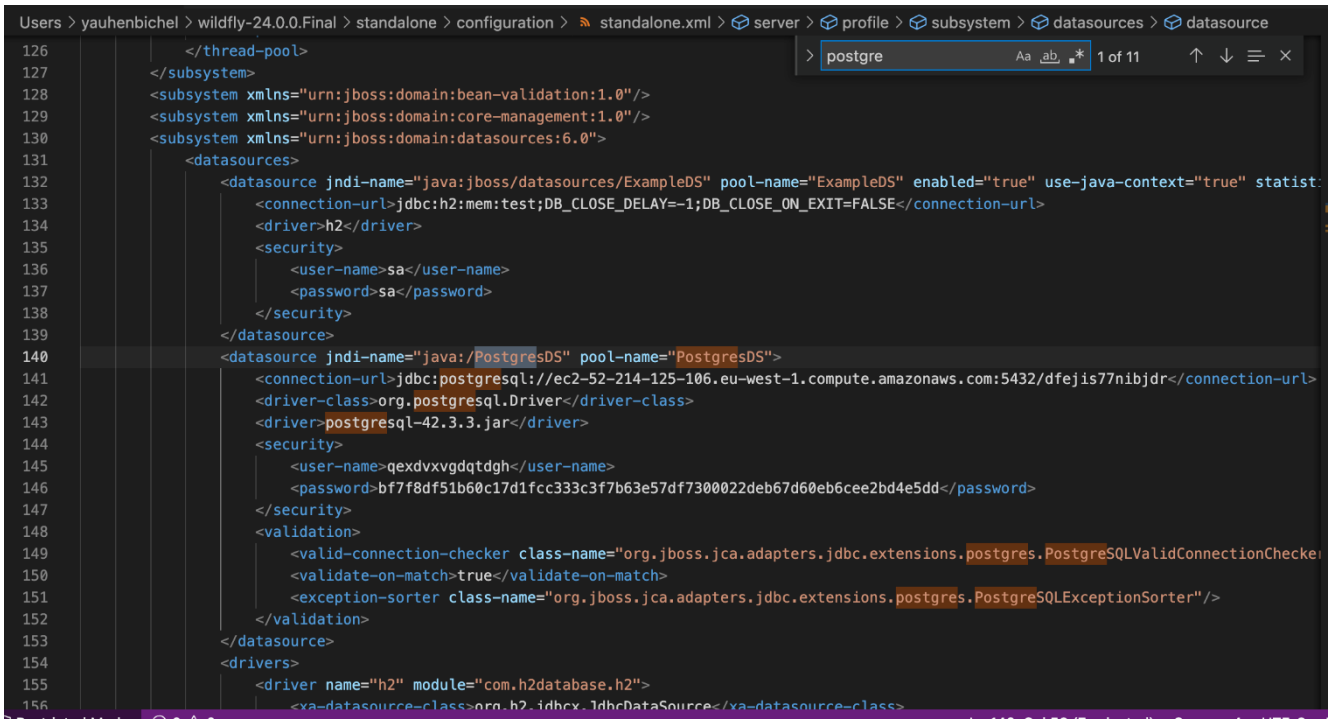


Figure 12 Database settings

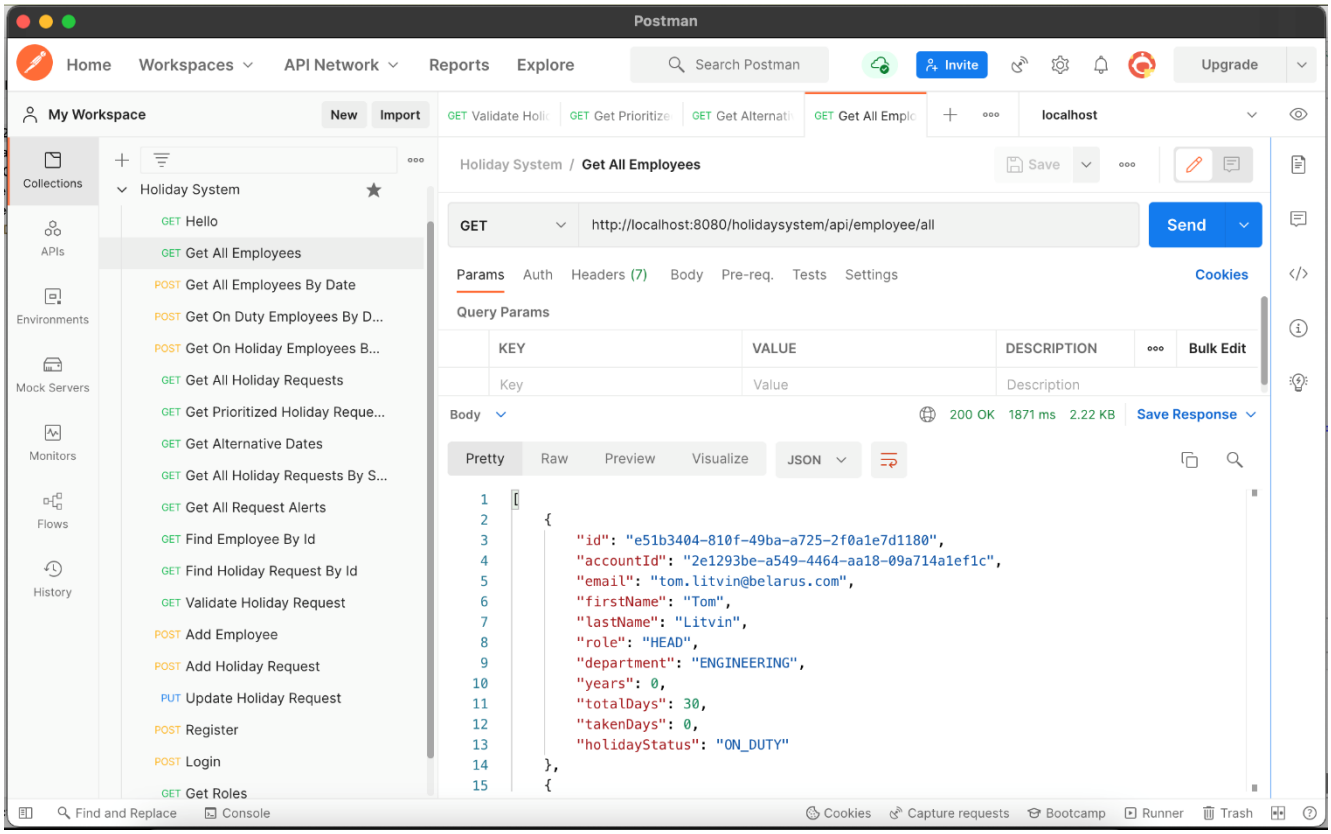


Figure 13 Holiday System API Request/Response collection

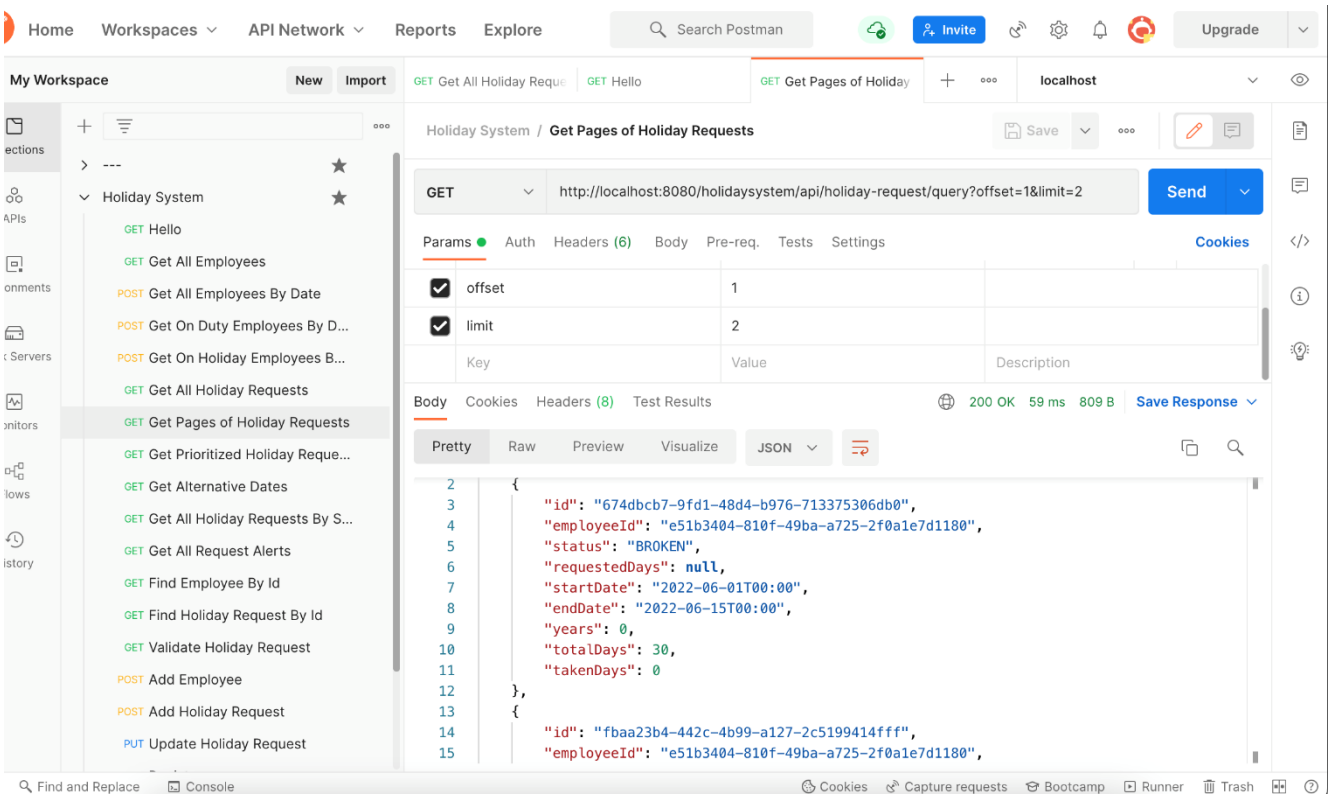


Figure 14 Pagination: query params: offset and limit sorted by created

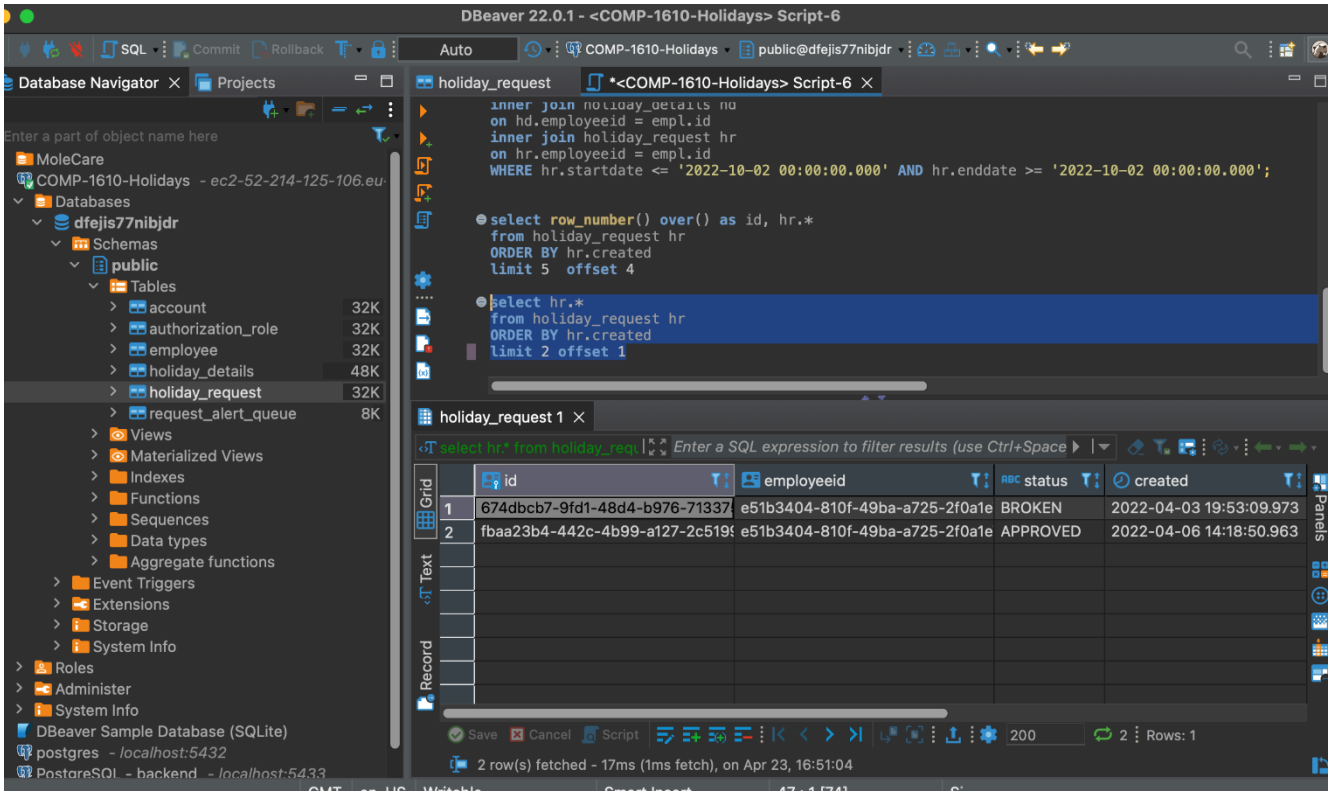


Figure 15 Pagination: database testing returns the same records

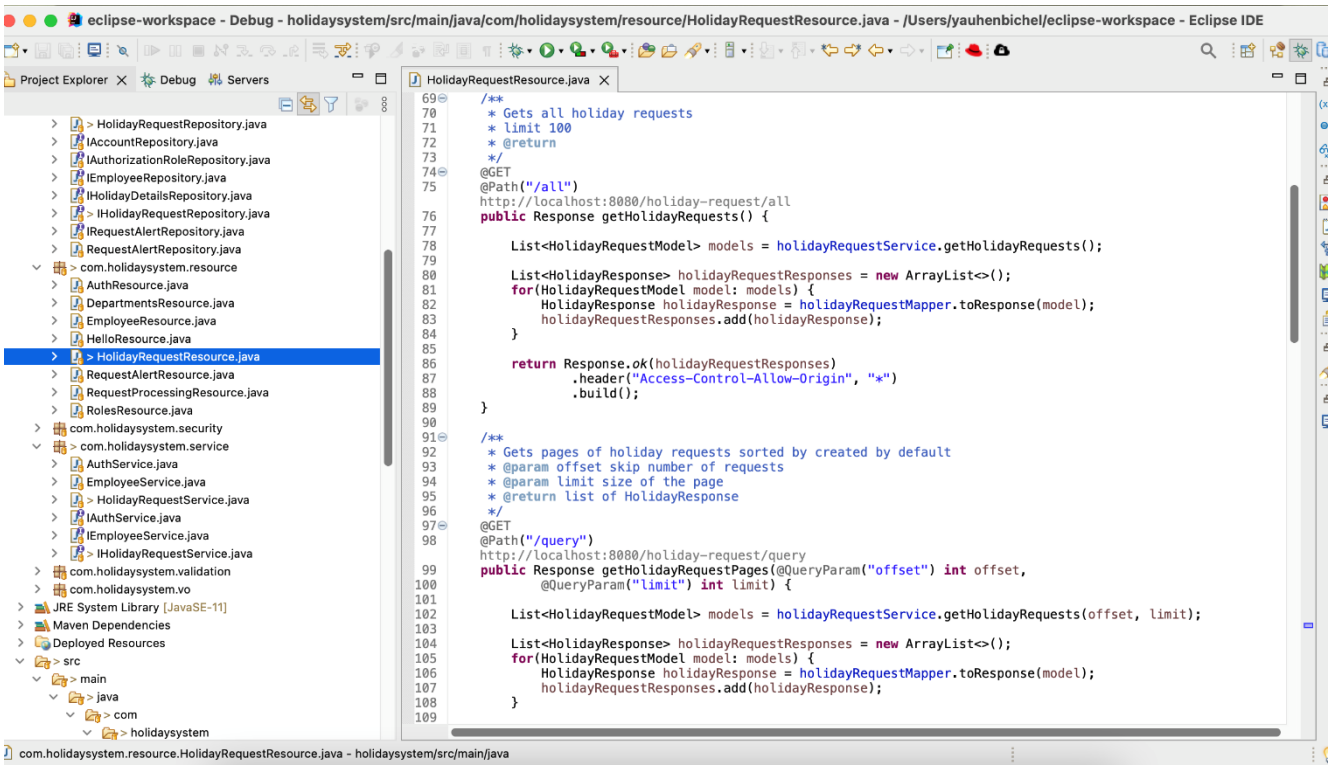


Figure 16 Paging implementing in REST API interface

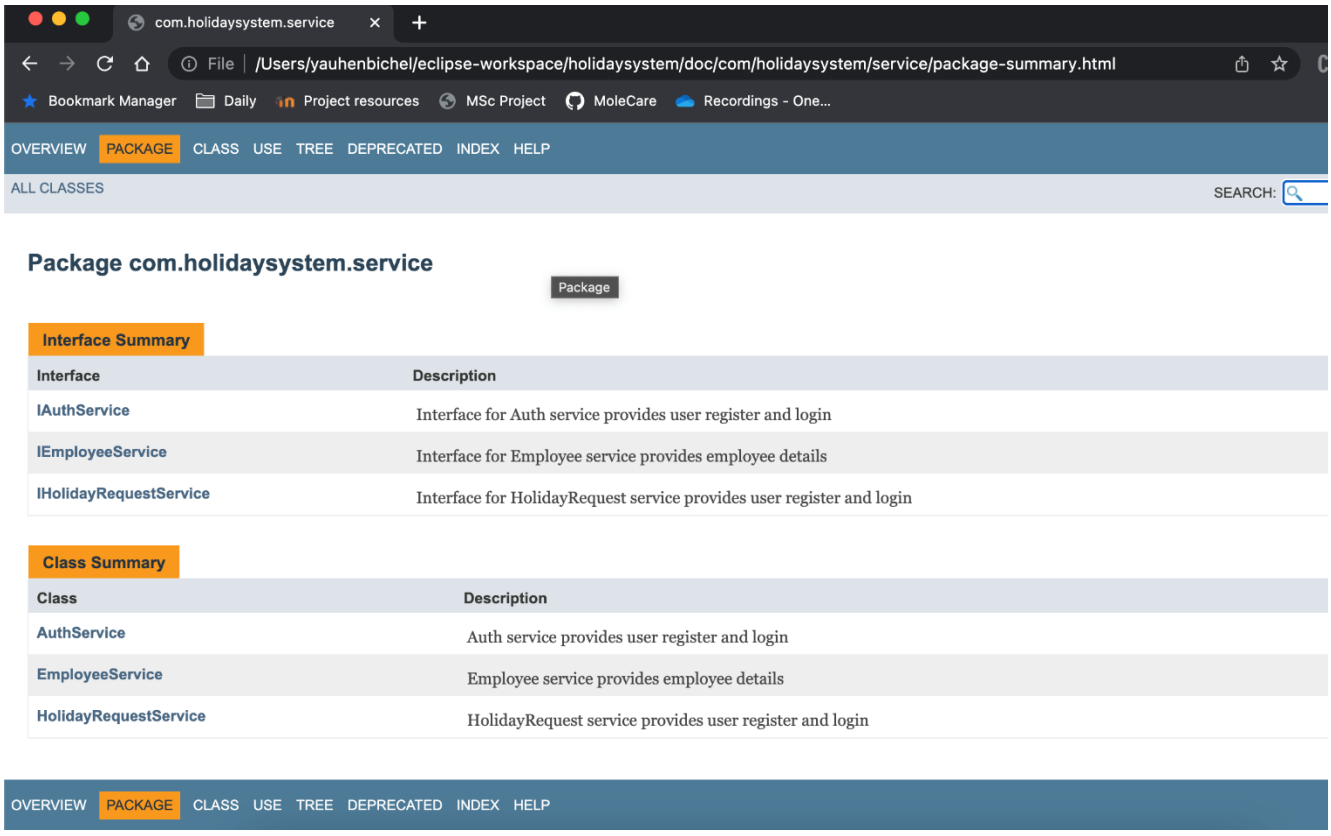


Figure 17 Javadoc Generated Documentation

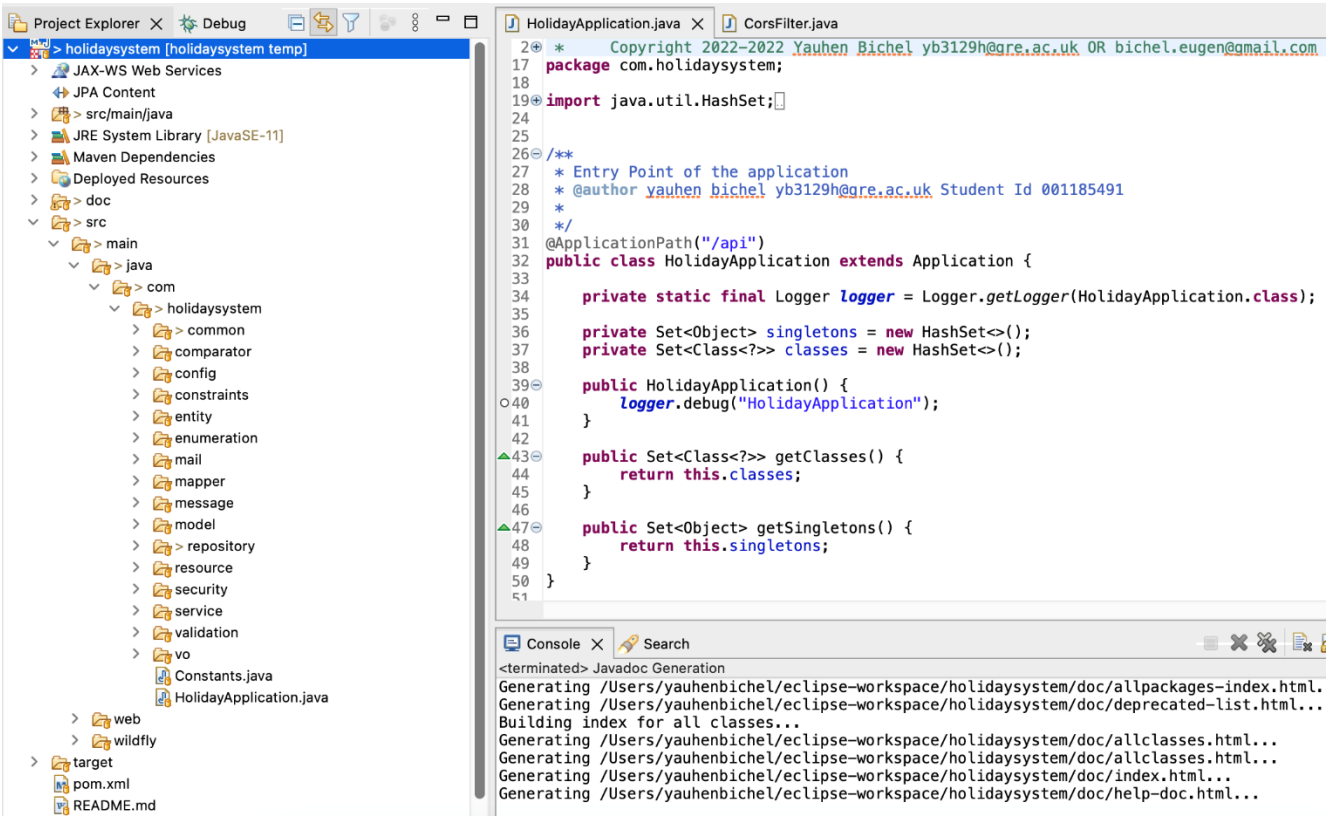


Figure 18 RESTful service project structure

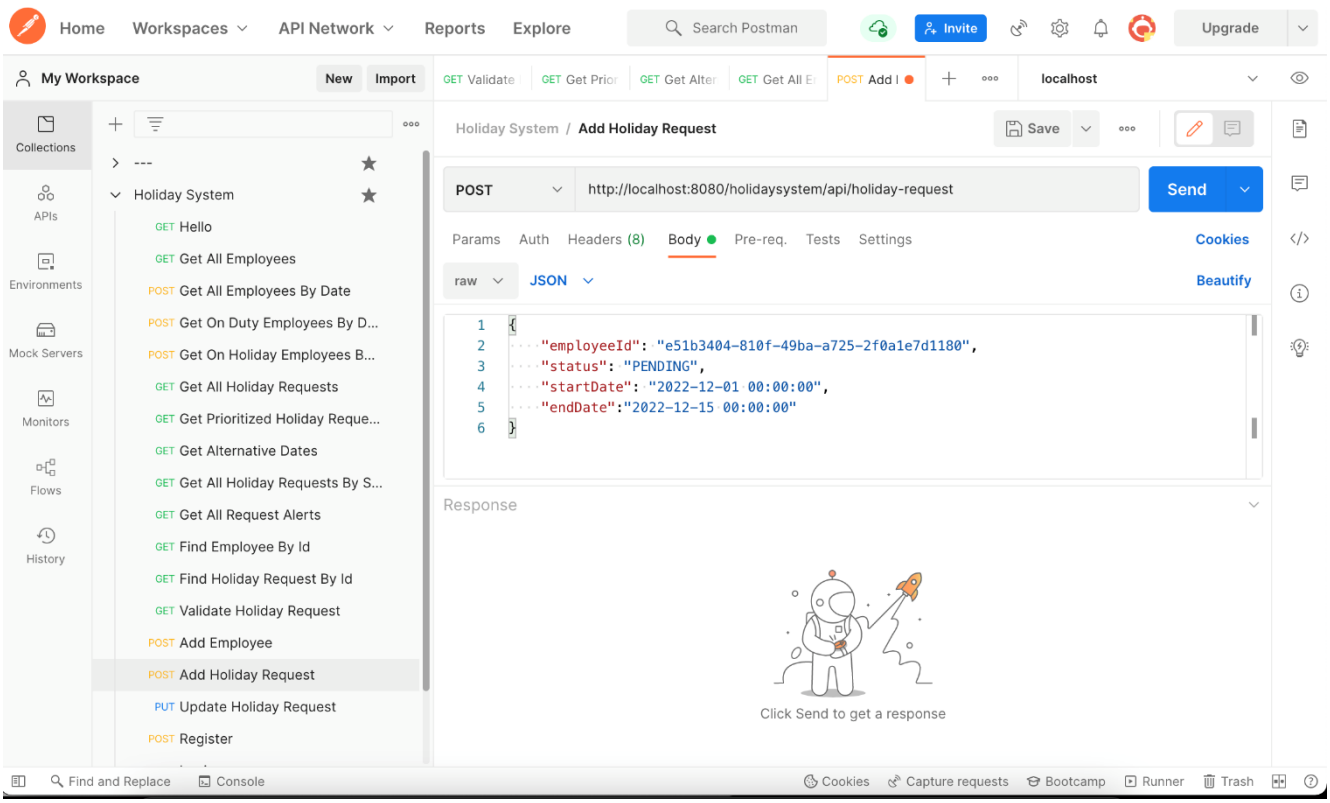


Figure 19 Add holiday request API request

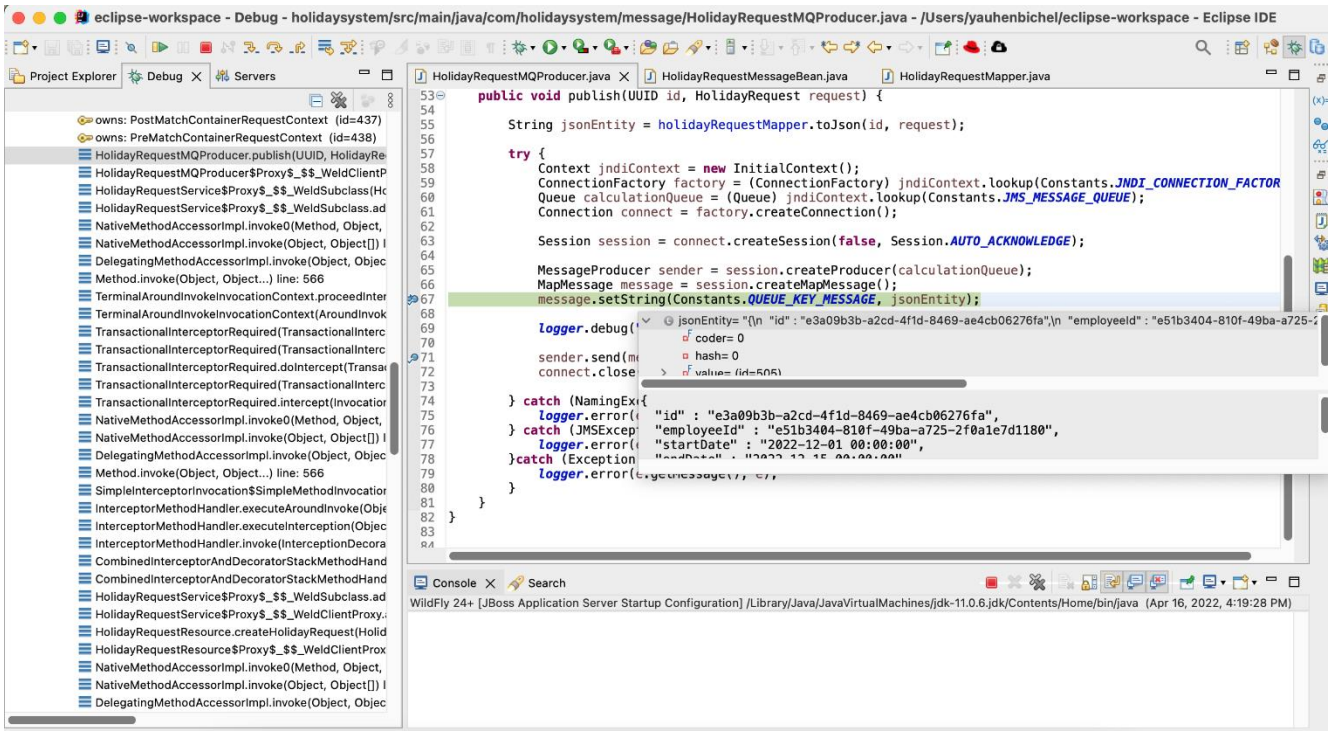


Figure 20 Put message to message queue

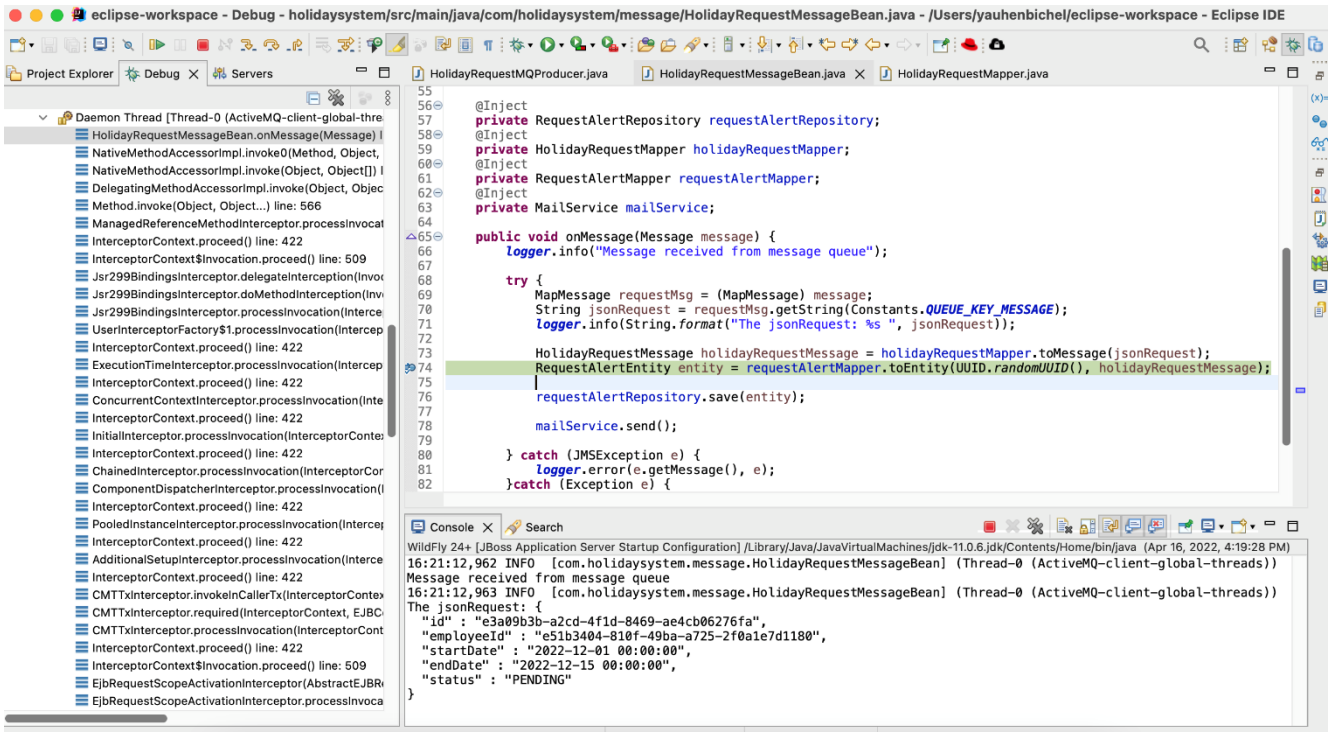


Figure 21 Receiving message from the message queue

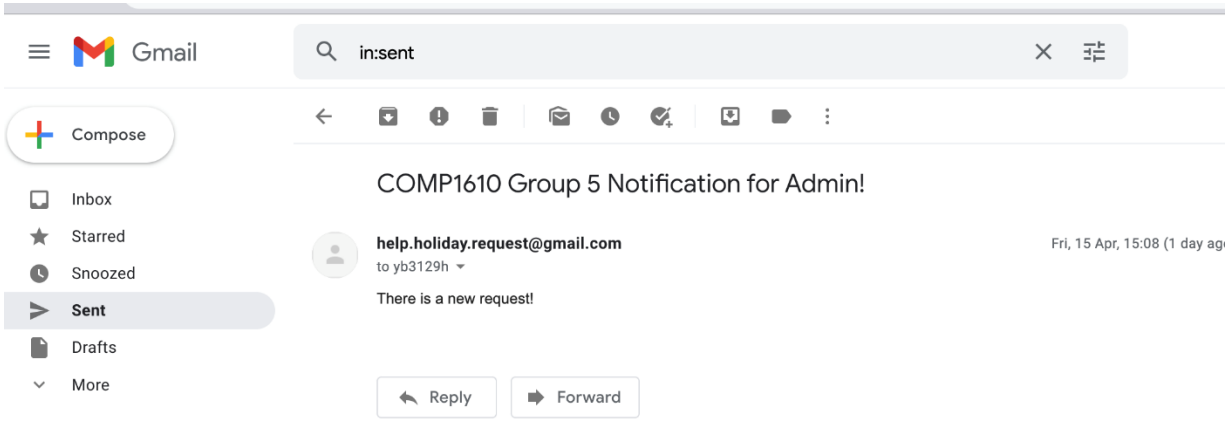


Figure 22 Holiday system app mail box to send a message to Admin. Sending by the application

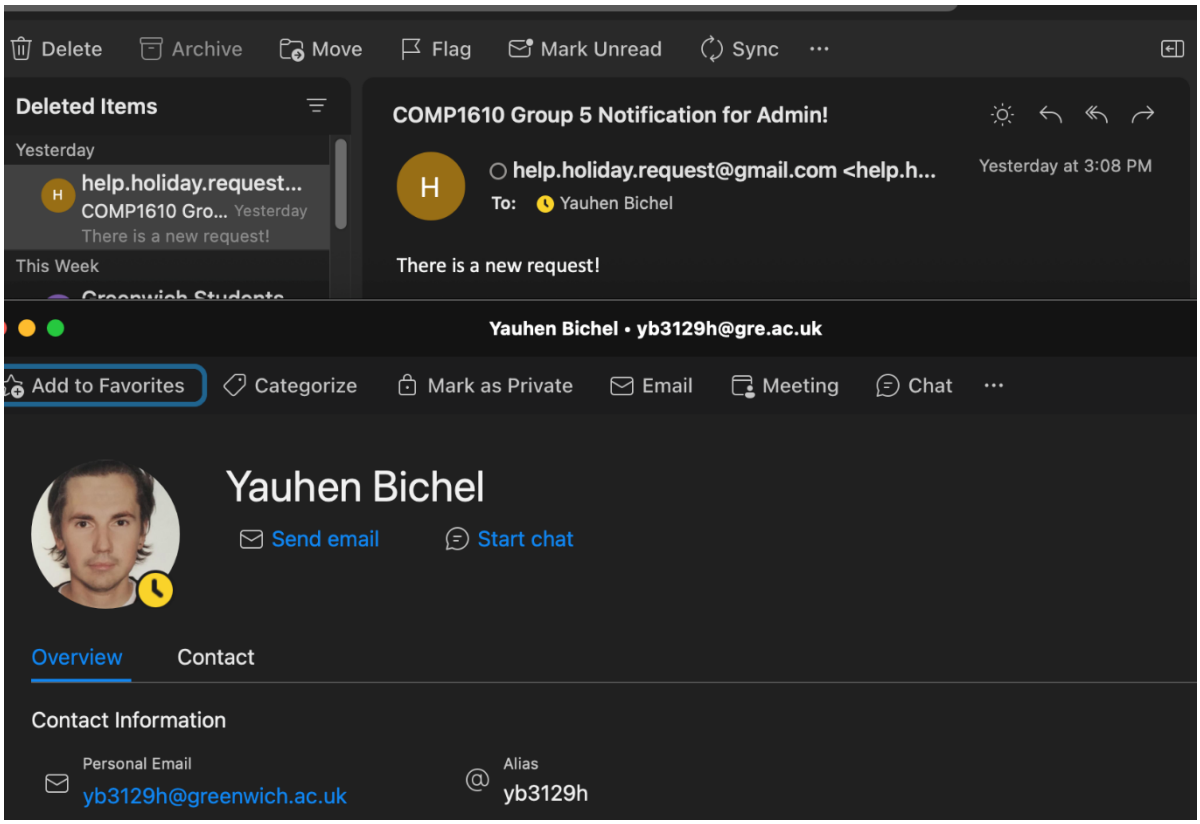


Figure 23 Admin receives the notification about new holiday request

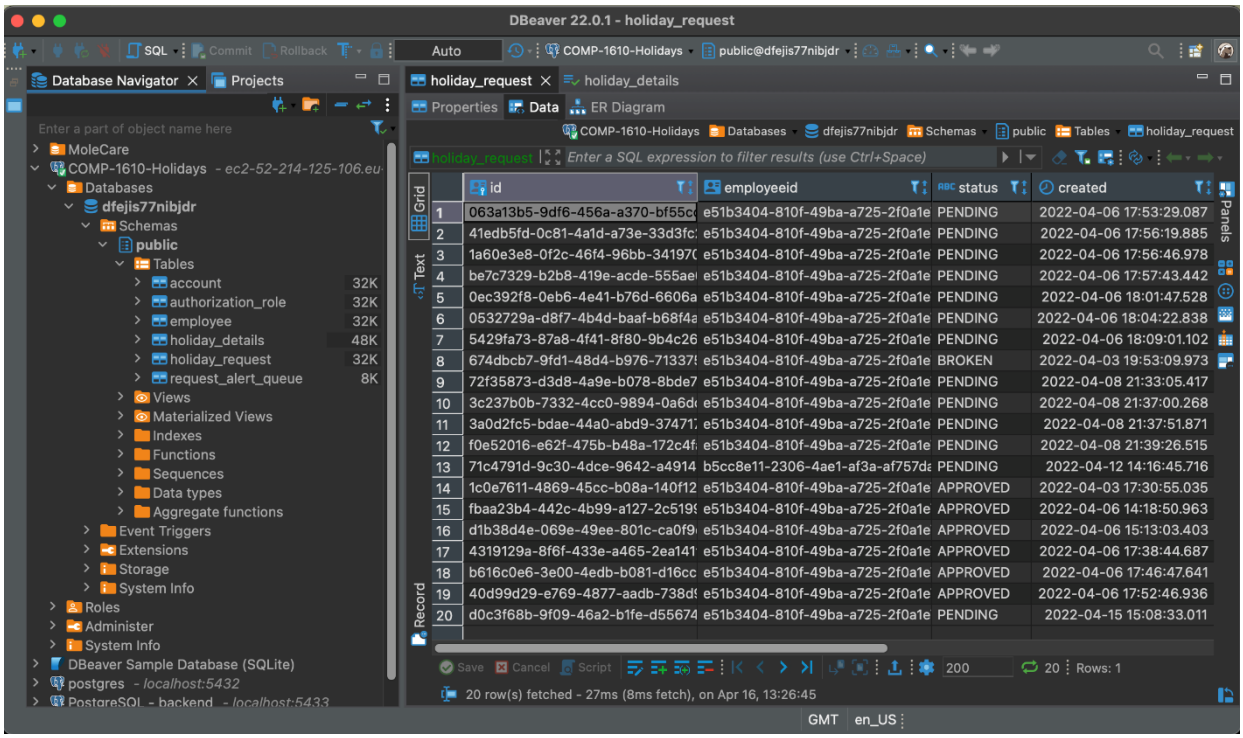


Figure 24 Prototype database, which should be implemented by design in debt tasks

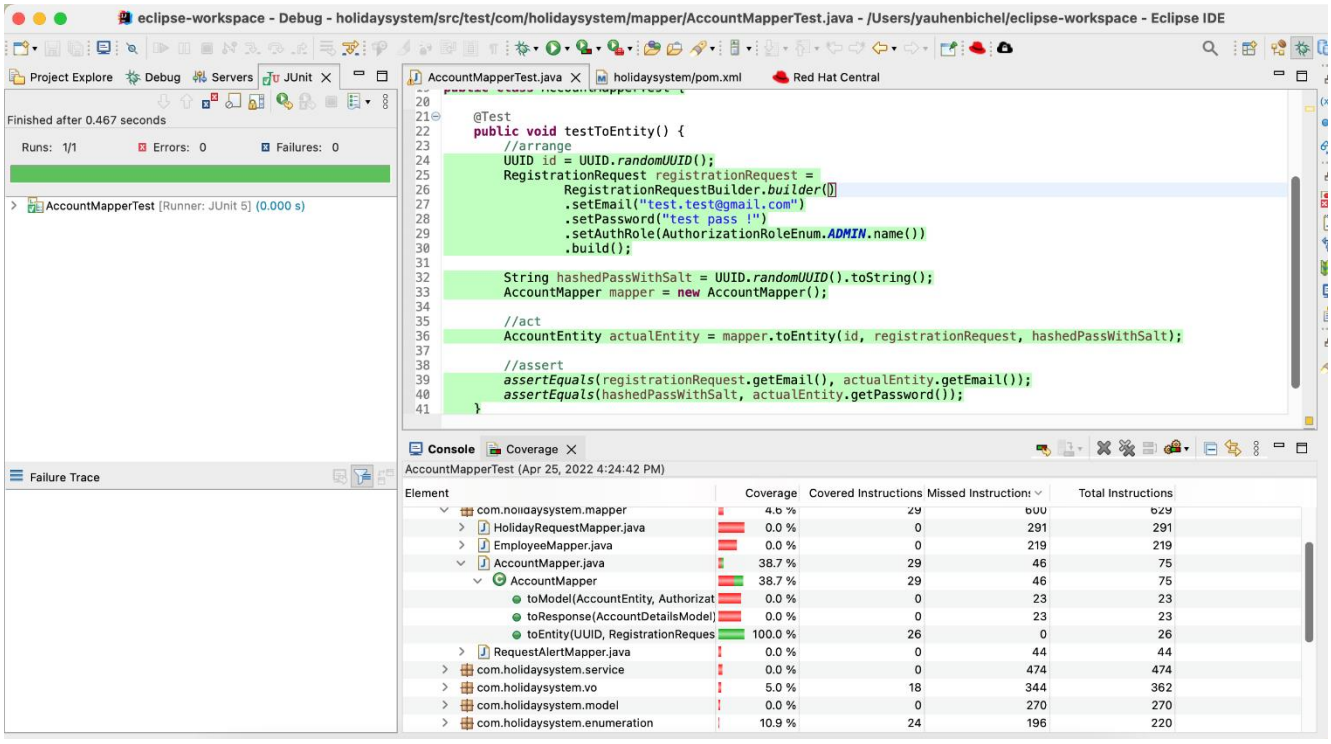


Figure 25 Unit tests and code coverage

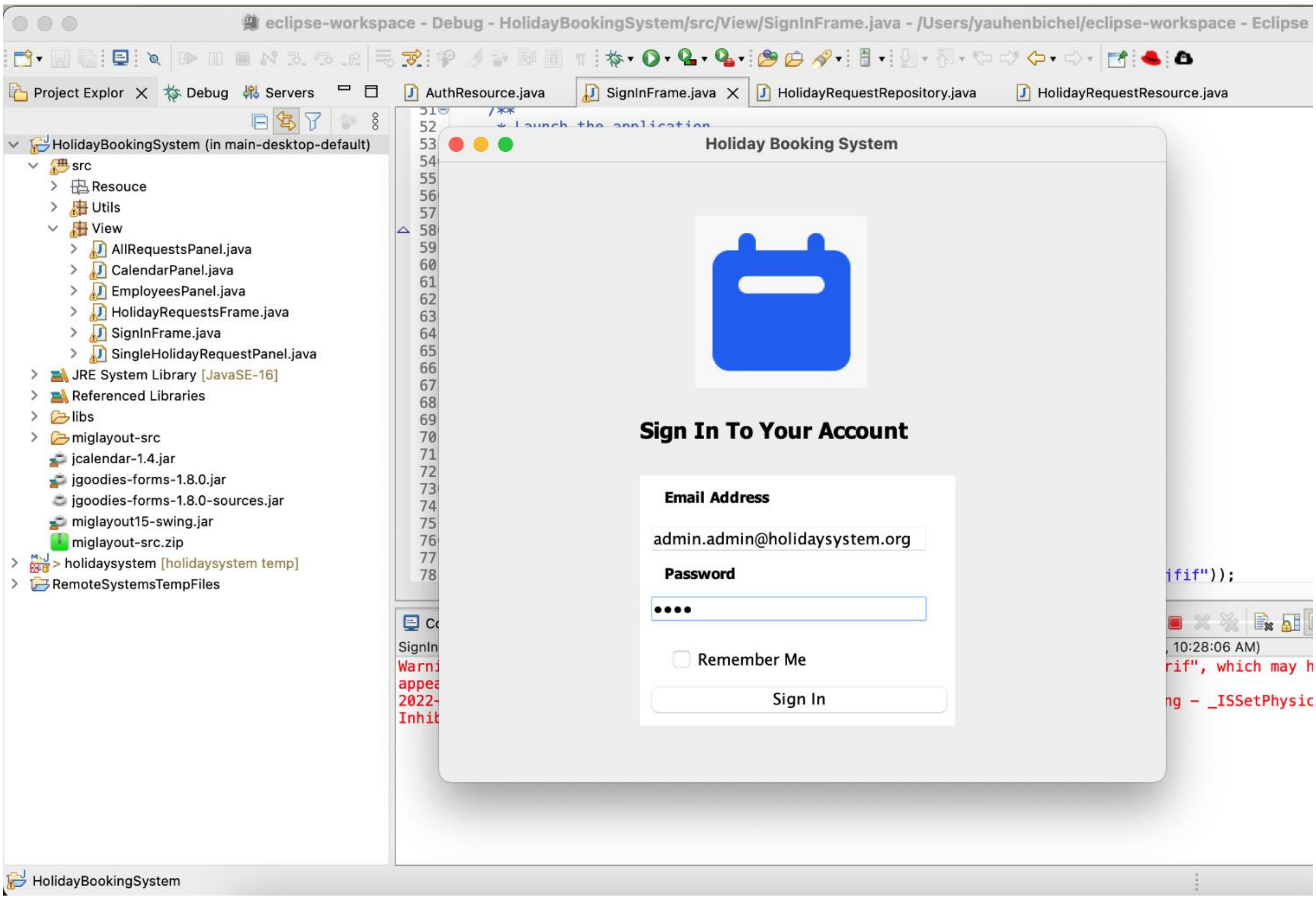


Figure 26 Desktop App Sign In form

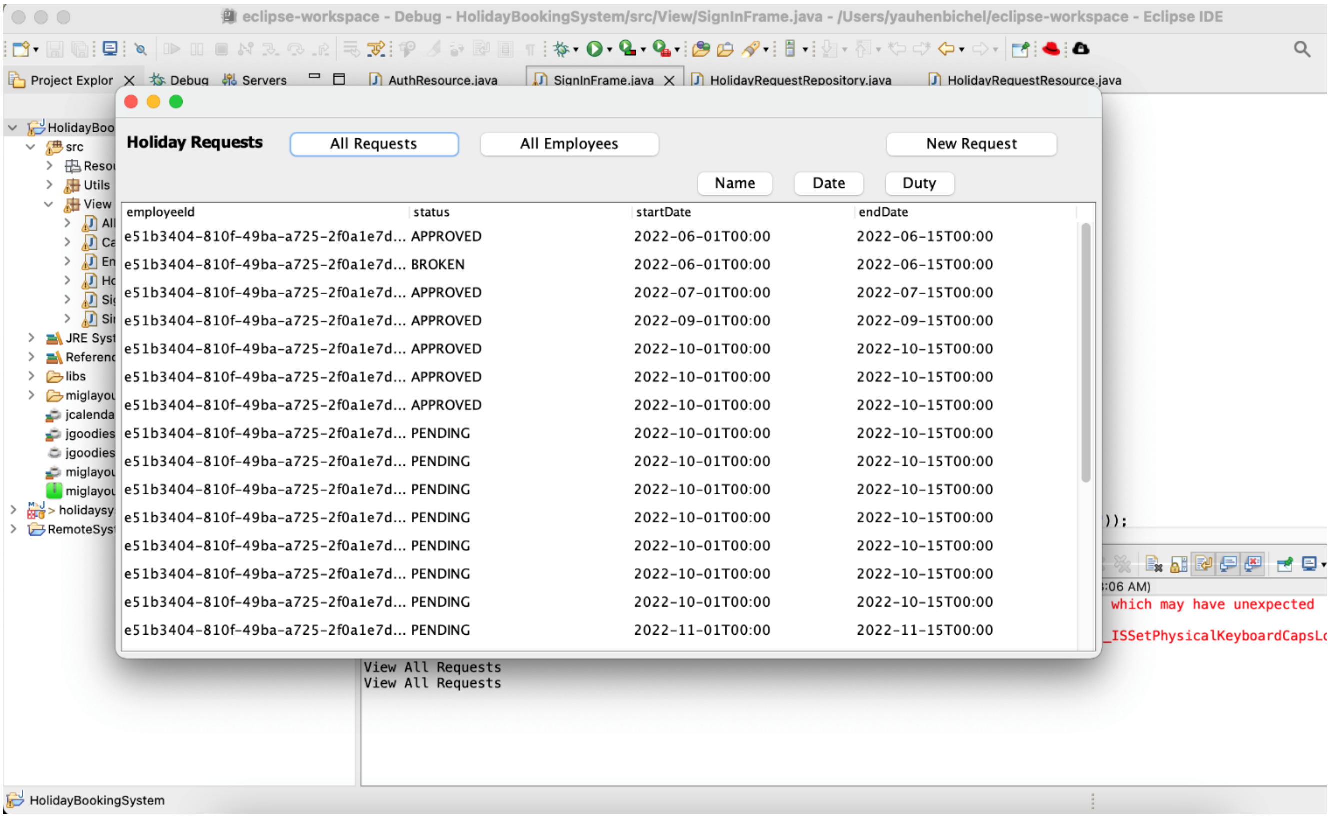


Figure 27 Desktop App Get Requests

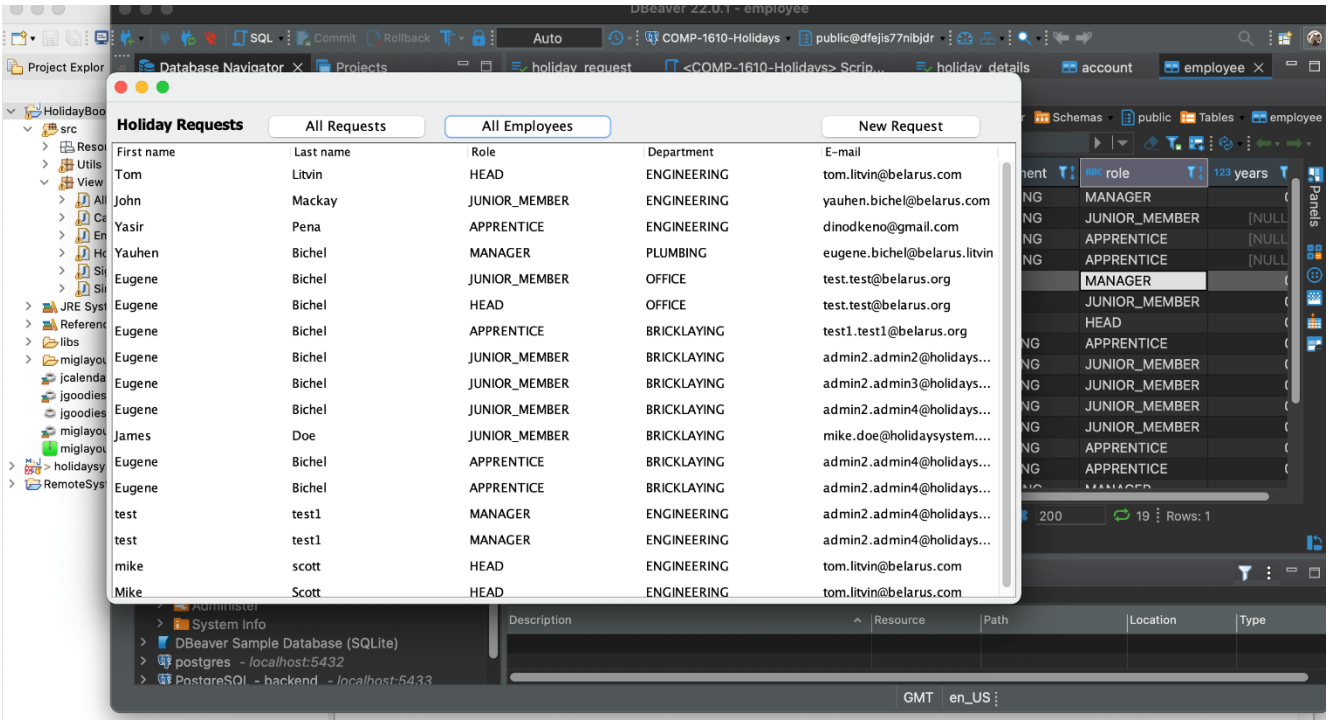


Figure 28 Desktop App: Get Employees

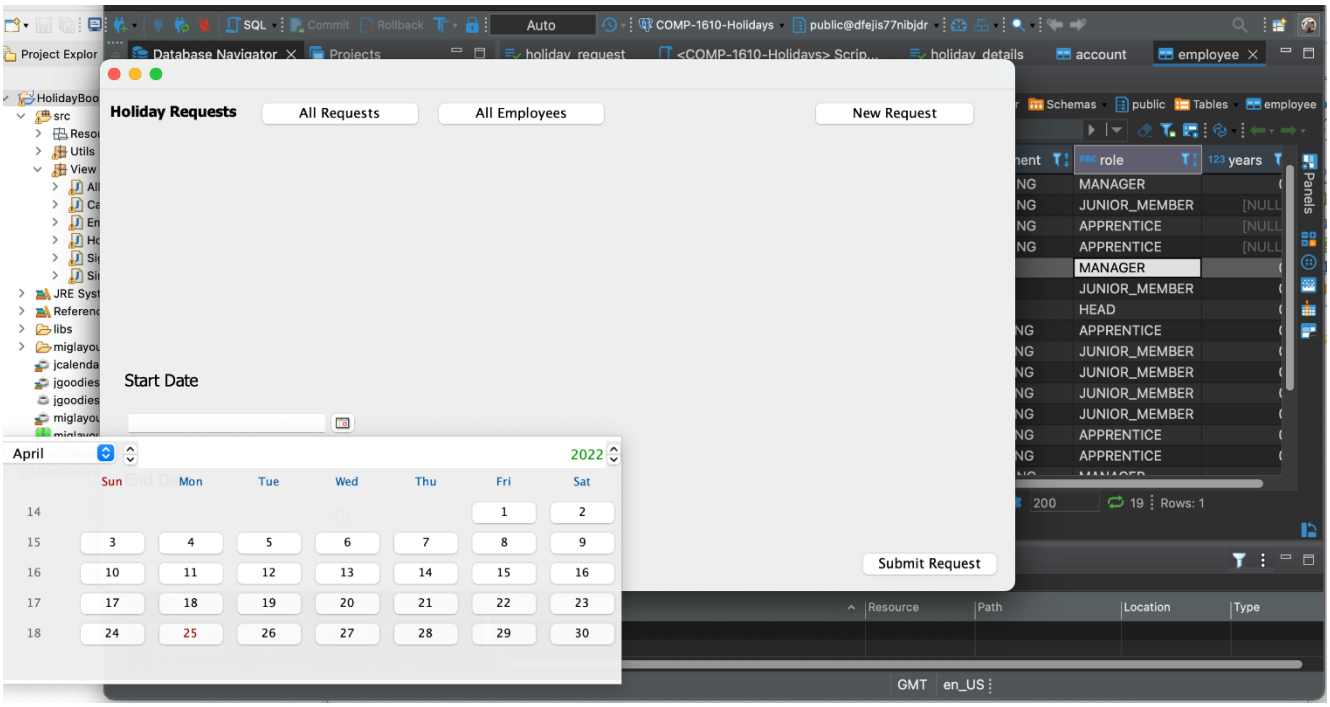


Figure 29 Desktop App: new holiday request

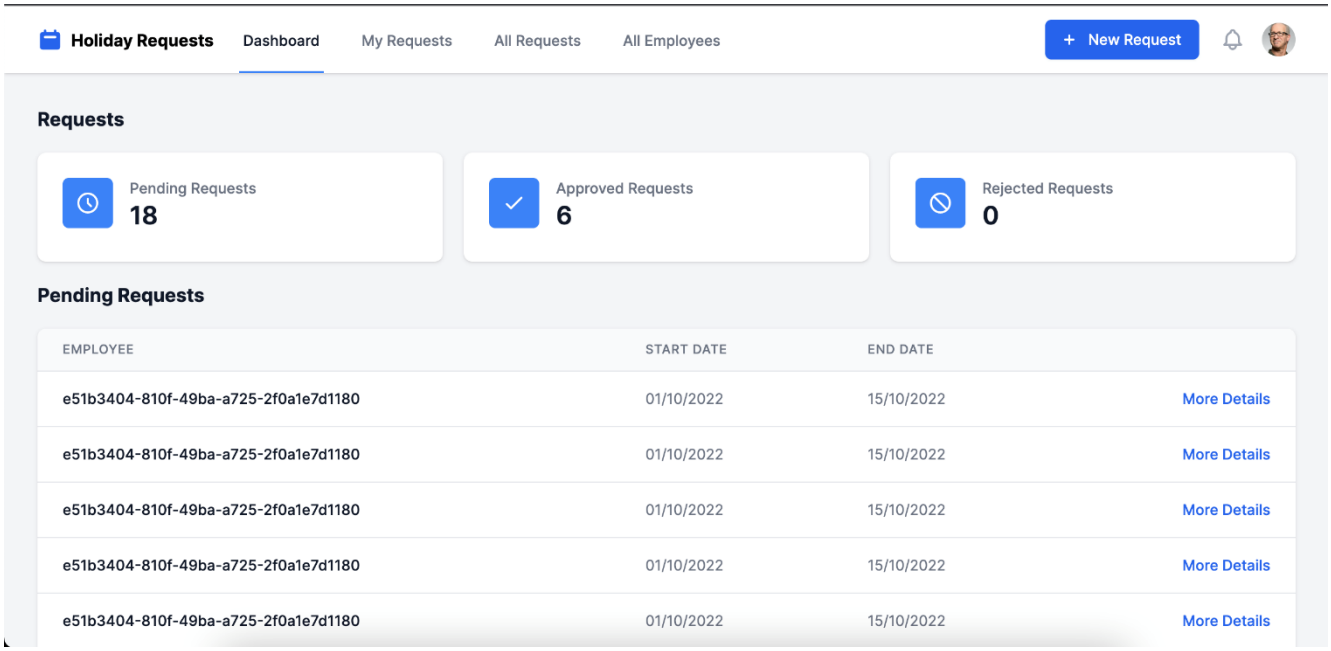


Figure 30 Web: dashboard

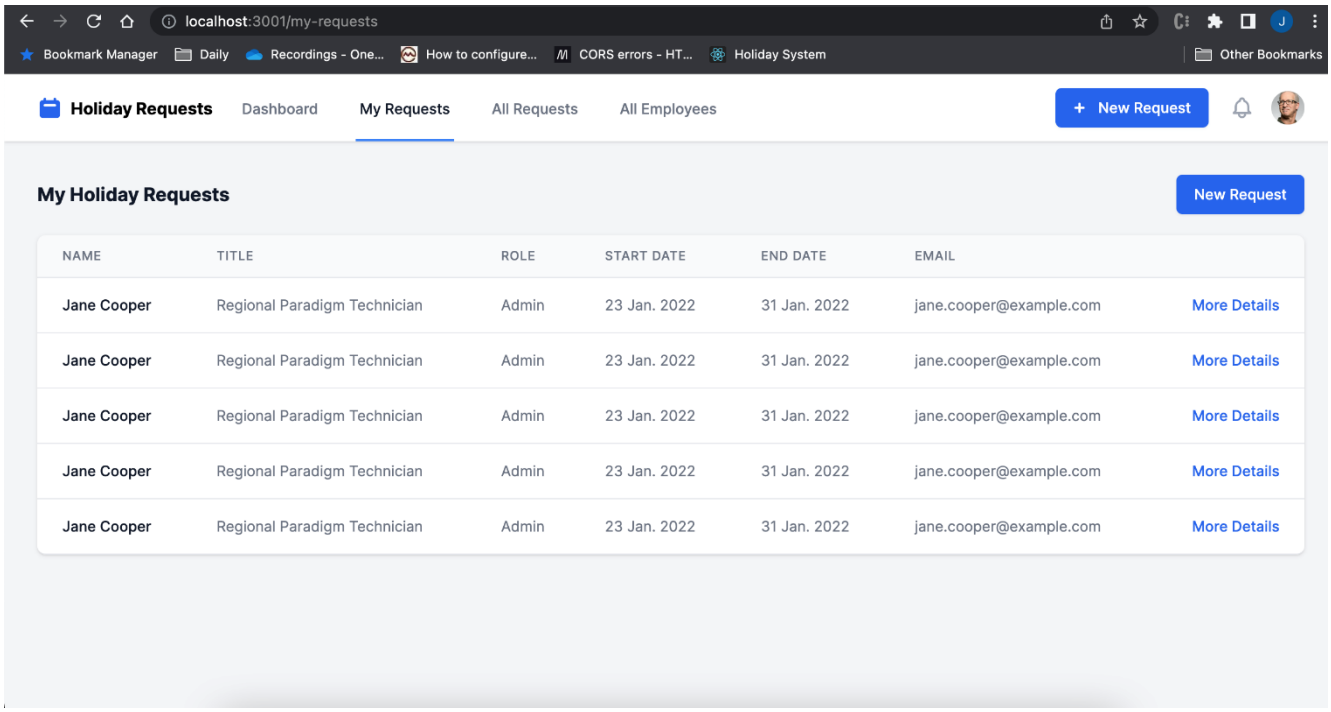


Figure 31 Employee requests view

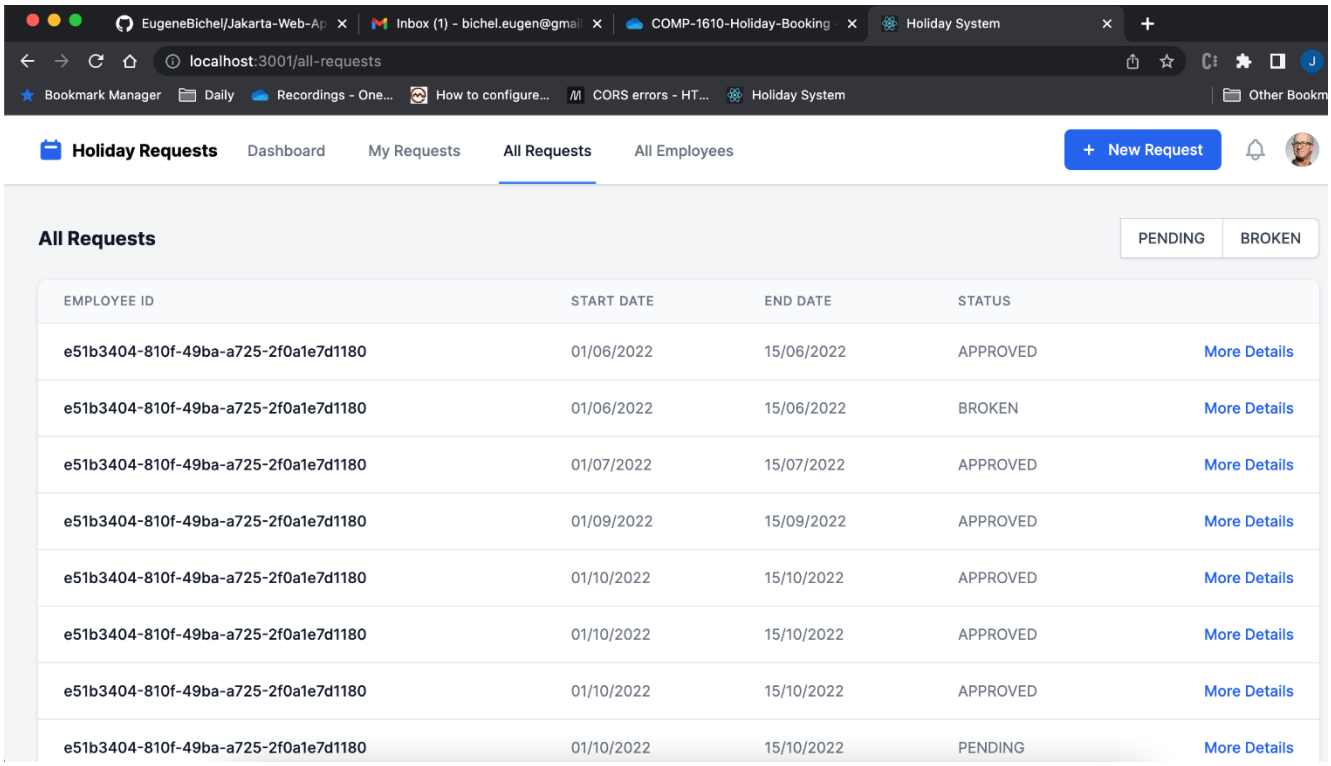


Figure 32 Requests without filters

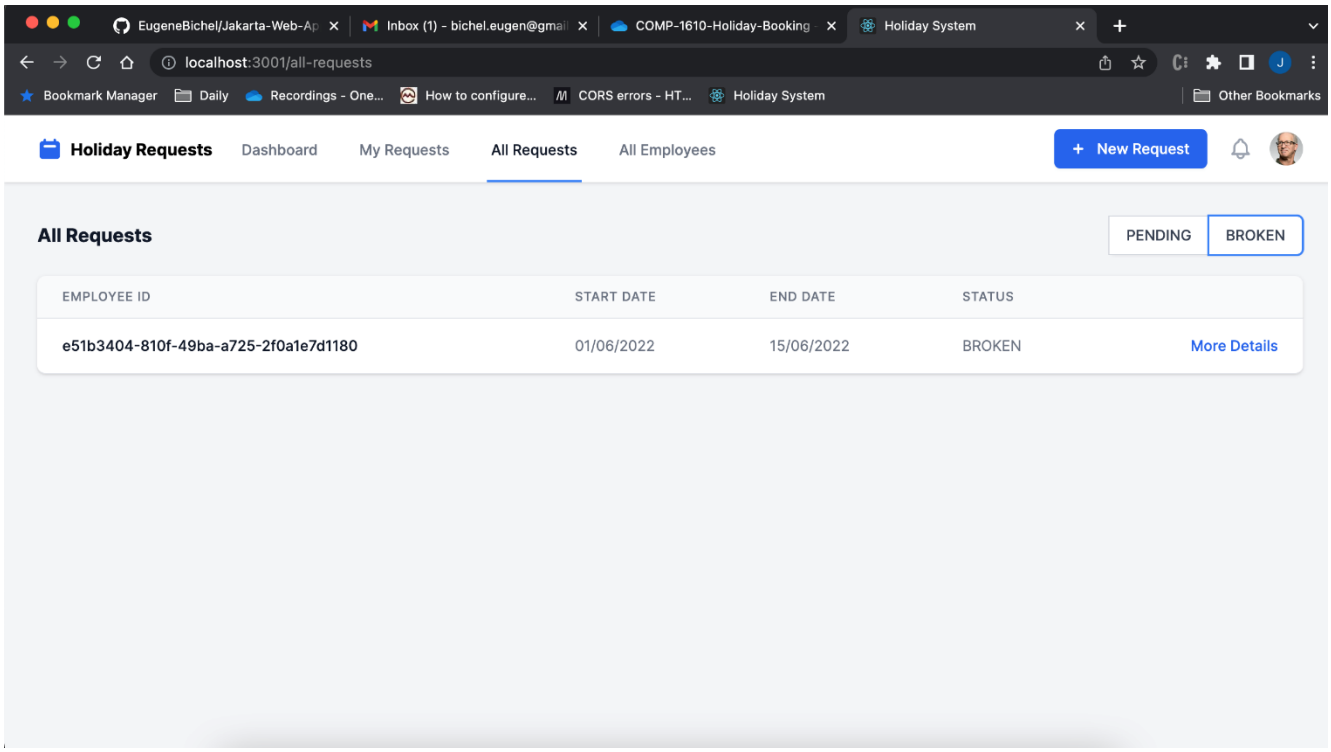


Figure 33 Requests with status BROKEN

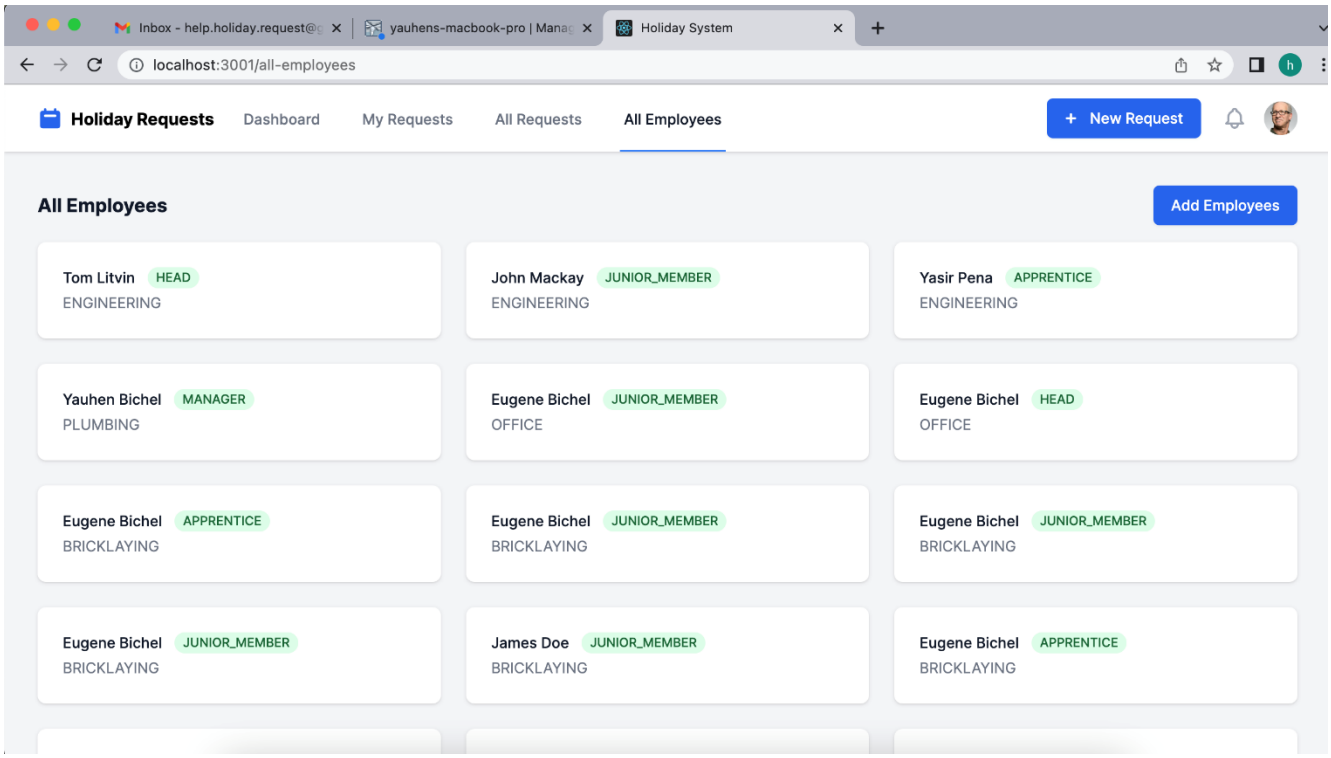


Figure 34 Web: all employees

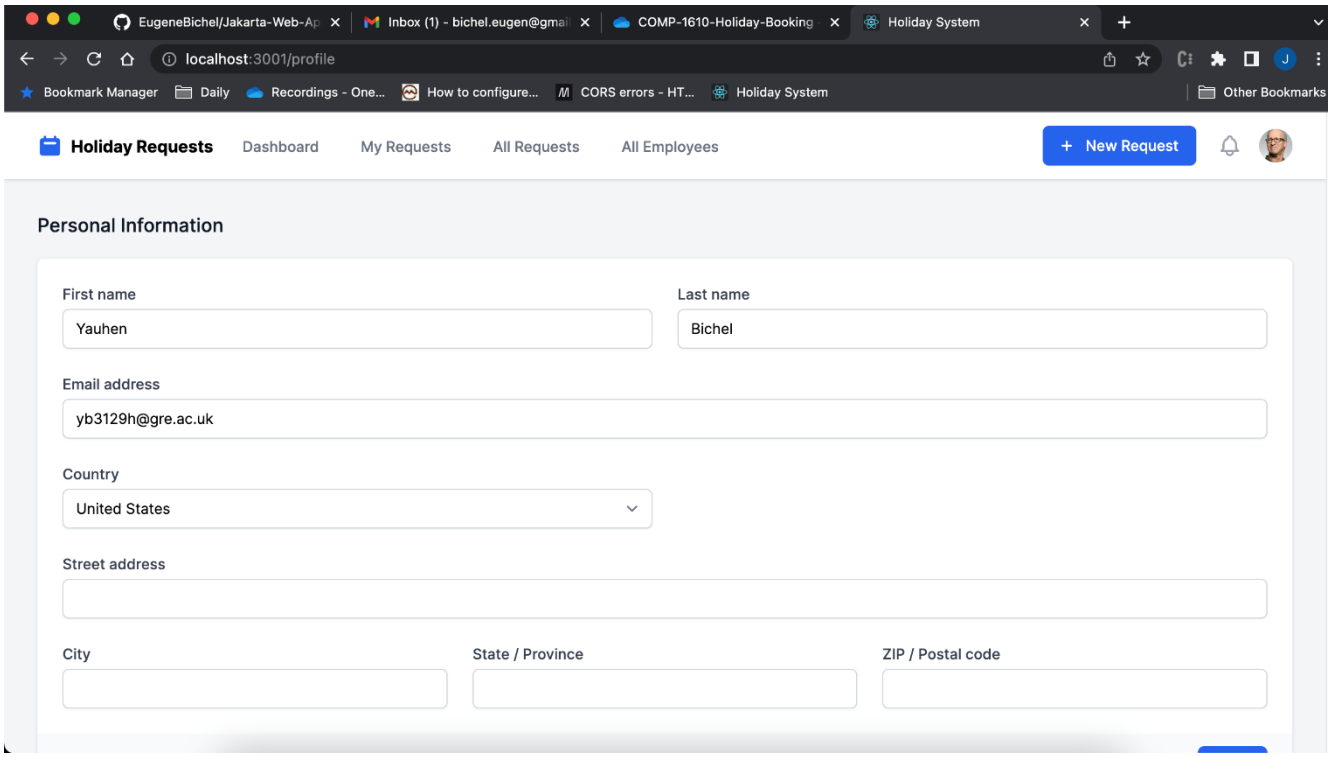


Figure 35 Personal view of employee

Evaluation

The solution for holiday booking system was designed and developed for 5 weeks. The product includes server part as RESTful service with PostgreSQL database, React web application as browser client and desktop application as one more client application for clients.

Design of server part was implemented by one student, Yauhen Bichel, the author of the report. Also, the server application configuration, REST API design, project structure design, code structure, implementing SOLID principles and GRASP patterns are designed and developed by Yauhen Bichel also. Apart from that, entity relationship diagram was designed and implemented by me. Moreover, I configured unit tests environment, start writing unit tests, run code coverage. Unfortunately, with reference to a short period of the module, I provided a demonstration how to run unit tests and use code coverage. Also, I help to implement function in web application, which uses React library, in particular, I implemented filter functionality, fixed dates format for REST service.

On the one hand, I think, we are, as the team, did the great job in a very small time period with quite big list of functionalities with supporting 2 different client applications as web application and desktop application. At the same time, I designed low coupled design of the RESTful service, designed names of resources, used DI for decreasing the dependency from implementations of interfaces, used repositories to separate the layers between domain entities and calls to database and business layer of the backend application. Moreover, I designed and realized message system, mail notification, logging system and providing human friendly messages from exception handlers. Also, interfaces methods and classes have Javadoc comments, and I generated Javadoc documentation. In addition to that, the Javadoc documentation, I created postman collection of REST requests/responses, which are added to zip file with source code of the backend application.

On the other hand, with reference to short period of time, which was given, the database schema contains simplified structure than entity relational diagram in the design. The exception handling does not cover all possible scenarios and it can be improved. It is important to mention, that I do not have enough time to write unit tests, add a few integration tests to verify communication between database and the application. The components do not cover all possible API requests, does not check all possible requests validation. Moreover, the communication with database was not analyzed using SQL query analyzer. I need to mention, that logging should be improved as well. As the result, the current solution is not stable, the users can see exception messages, server errors, database can reach inconsistent state. The application does not have retry strategy for failed connections or weak connection between application server and database server. The logs can have user critical data, which should be tested and fixed, also some components can have low covering by logs, and it should be improved as well.

Apart from backend RESTful service, the client applications are in prototype state too. They are not tested well, and they do not have good structure. They do not have any procedure to repeat requests when the connection is weak.

Also, I want to mention, that the significant part of the configuration, design and developed was done by me, Yauhen Bichel. I configured, designed and developed Jakarta EE application by myself. Other students designed and developed React application and desktop application, also they added integration to RESTful service with my help. Moreover, I helped to setup the WildFly application server and the backend application in they local machines. I created the GitHub repository, where I consistency make pull requests, and each member of the team can see the changes. Unfortunately, other members of the team do not have the time and some experience to do the same. As result, the gap of knowledge and experience between students cannot help in development of the big systems. I have to mention, that we had weekly 1-2 hours meetings during the almost 3 months.

Taking everything into account, the team worked hard, and we implemented the functionalities from requirements. With reference to the fact, that the system is a prototype, in my opinion, the provided solution is quite good. At the same time, the application has a long list of technical debt and secure and stability improvement tasks, which should be designed and developed for starting to use the application in testing with external testers.

3. PART B (COMPLETED INDIVIDUALLY) (20%):

1. **Research (approx. 600 words):** Jakarta EE makes it possible to create enterprise applications. Carry out some research and critically discuss what other technologies and frameworks are available and how they compare to Jakarta EE.
2. **Individual Reflection (approx. 300 words)**: Please include a reflection of your role within the team and discuss lessons learnt. What you think went well and what you think could have been improved and how.

1. Research

Jakarta EE is formerly known as Java EE or J2EE. “Java Platform Enterprise Edition (Java EE) is an umbrella standard for Java's enterprise computing facilities”. (docs.oracle.com, 2022). So, as an umbrella of standards, there is a set of different specifications, which allows to keep compatibility of different libraries, tools and frameworks. For example, most of applications, in particular, enterprise applications have security functionality, transactions, logging libraries and logging analysis modules, often message queue systems for supporting asynchronous and robust behaviour, JDBC API to connect to different databases, persistence API for object relational mapping, also all enterprise applications require validation, authentication, authorization and other subsystems, which are commonly used in

applications. There are many frameworks, libraries, different platforms, which should be comparable for writing difficult enterprise applications. With reference to the fact, that most of applications can be divided by tiers and their have common components and subsystems, Jakarta EE provides a set of standards for different systems, which used for developing enterprise applications using Java. That is why, when the application is built using Jakarta EE model, using frameworks, libraries and tools, which are support a set of standards from Java EE, then the application can work in different environment without changes in source code.

During my prototype development, I used WildFly, which supports Jakarta EE standards. WildFly allows to use different subsystems, as fact, I used data source subsystem to add JDBC driver and set connection settings, messaging subsystem to add queue and topic. For example, WildFly provides RESTEasy implementation of JAX-RS standard for building REST services, and ActiveMQ is messaging system in WildFly by default.

There are many different technologies and frameworks, which can be used for developing Java applications and enterprise applications. However, one of the main requirements for most of enterprise applications is compatibility with standards, and there is Jakarta EE standards for Java enterprise applications. WildFly is one of the application servers, which is compatible with Jakarta EE standards. Also, GlassFish server is one of other examples of Jakarta EE compatible application server. (javaee.github.io/glassfish, 2022). WildFly and GlassFish manage Java EE applications. Moreover, Oracle WebLogic Server is one of popular application servers for managing and deploying Jakarta EE applications.

One of the alternatives for using applications servers as WildFly is Spring framework. Spring provides components for all required subsystems in enterprise applications. As a developer, we can find all components in Spring as in Jakarta EE applications. Spring provides security, batch component and scheduling services, persistence API and other modules and components. At the same time, Spring provides their own implementations as Spring MVC for REST services, when Jakarta EE compatible applications use JAX-RS compatible implementations as RESTEasy and Jersey. Also, when Jakarta EE applications use EJB for implementing business logic in enterprise applications, Spring framework provides Spring Beans like Service, Component, Repository. The idea of them are the same using interfaces and providing different implementation. Apart of it, Jakarta EE applications use message driven beans for message systems, when Spring suggests Spring JMS.

Taking everything into account, there are many different Jakarta EE applications servers for managing and deploying Java applications, also as an alternative for traditional Jakarta EE servers, Java enterprise applications can be developed using Spring framework. Some of the main criteria for choosing the application server and framework is documentation, professional community and Jakarta EE standards compatibility. At the same time, Spring framework provides some compatibility, for

example while Java EE uses CDI contexts and dependency injection standards, Spring provides a dependency injection container. But CDI is an option for DI in Spring.

2. Individual Reflection

Design for the Jakarta EE application was made by me. I separated functionality into logical components, I defined user flows, I created entity relationship diagram.

Also, I setup WildFly application server, configured relational database for the application. Helped to setup Eclipse IDE and WildFly application server for other members of the team.

Apart from it, I setup datasource, message system, mail, created gmail account for the application.

After that, I created README.md file with settings and added configured standalone file with jdbc postgresql driver jar file into the application folder.

I designed and developed from scratch Jakarta EE application. For all designed and implemented public API, I provided examples of REST requests and responses in postman collection using the following tool https://www.postman.com/downloads.

Moreover, I teach all other members of the group how to think about design, what structure of the code should be, how to write code, how to provide low coupling with high cohesion, implement unit tests.

During the collaboration with my team, I scheduled the library meetings, discussed our time management, tried to fix different issues like CORS errors.

Regarding the learnt lessons, I would like to mention that it was my first experience of developing Jakarta EE applications, and I learnt many new tools and servers, and how setup application server for enterprise applications, which supports Jakarta EE standards. With reference to my experience with Spring framework, I can compare development of Java applications using for example WildFly application server and Spring framework.

Taking everything into account, I learnt WildFly application server, Jakarta EE standards, configuration datasource, message subsystem, mail subsystem, logging, debug application in application servers. However, I found very difficult to collaborate in the team in very short time periods in two quite big projects. I think, the developed product, which includes quite big backend part with separated web application and desktop application requires at least 6 months of development. That is why, I found the module is quite difficult because of very short time period for developing the product in a team with very different level of knowledge and experience. For example, I designed and developed a full RESTful service and configured the application server by myself. And, in my opinion, it is not related with management or time management, the problem is very different level of students

in a very short time period. Also, the implemented by me prototype does not mirror my design fully. During the implantation I simplified the domain model and functionality to provide worked prototype.

I am glad to learn WildFly application server and developing enterprise applications using Jakarta EE specifications.

4. COURSEWORK CONTRIBUTION (COMPLETED AS A GROUP)

In percentage, please indicate the work contribution of each member. This should be agreed by all group members**. The total of all members work must add to 100%**

Team member name	Student ID	individual overall work contribution (%)	Signature	Note
Student: Yauhen Bichel	001185491	30	Yauhen Bichel	- A full design of the system and backend part design in particular. - WildFly configuration and writing steps for setup. Help with setup for team members. - Design and develop Jakarta EE application using WildFly server and its subsystems. - Selecting database, adding tables to database, configuration Heroku for the application. - Providing API requests; - Implement filter functionality in client side; - Design and implement Pagination

				- Design and implement message queue system and mail service - Design and implement validation, exception handling, logging and added Javadoc comments - Generated Javadoc documentation
Student: Ivan Huliyev	001181952	20	Ivan Huliyev	
Student: Yahya Chahine	001178039	25	Yahya Chahine	
Student: Nurkaiyr Yedige	001196862		Nurkaiyr Yedige	
		Total 100%		

Describe each task you performed	Student: 001185491 Yauhen Bichel work contribution in %	Student: 001181952 Ivan Huliyev work contribution in %	Student: 001178039 Yahya Chahine work contribution in %	Student : 001196862 Nurkaiyr Yedige work contribution in %	**total**

Feature A:	70	10	10	10	100%
Feature B:	40	20	20	20	100%
Feature C:	60	10	20	10	100%
Feature D:	40	10	10	40	100%
Feature E:	25	25	25	25	100%
Feature F:	70	10	10	10	100%
Feature G:	40	10	30	20	100%

5. MANUAL DOCUMENT

This section explains how your prototype should be run with clear instructions

There are 3 separated program modules: RESTful service, React web application and desktop application.

RESTful service contains README.md file with the following steps:

- how to setup WildFly server;
- how to run the application in the application server with settings for standalone from the folder with the application source code;
- where database is deployed and how to connect to the PostgreSQL database for the web service.
- Mail box of the application, which is used for sending the message.

1. Firstly, please setup WildFly application server, which supports Jakarta EE standards.
    a. Download the WildFly 26.0.1.Final zip file here https://www.wildfly.org/downloads/
    b. Copy the unzipped folder to the place in disk, where the system contains application servers.

c. Rename wildfly-26.0.01.Final folder to wildfly-24.0.0.Final for supporting by Eclipse

2. After that, please install Eclipse and run the application in Eclipse IDE.
   a. Download the Eclipse https://www.eclipse.org/downloads/
   b. Download the Jakarta EE from the group 5 github repository https://github.com/COMP-1610-cw-team/backend . Also, the zip folder can be downloaded from Moodle submission folder in Moodle COMP-1610 https://moodlecurrent.gre.ac.uk/course/view.php?id=70195
   c. Open Eclipse IDE and click File -> Open Projects from File System
   d. Open folder with the holidaysystem project
   e. Find standalone.xml file with settings for WildFly application server for the application in src->main->wildfly folder.

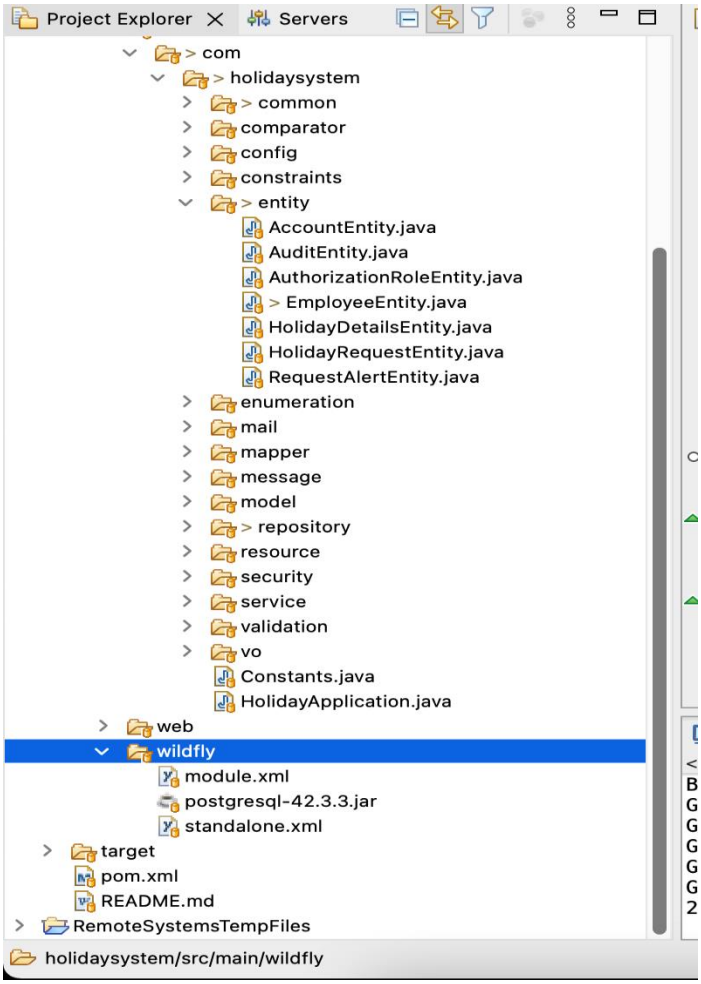
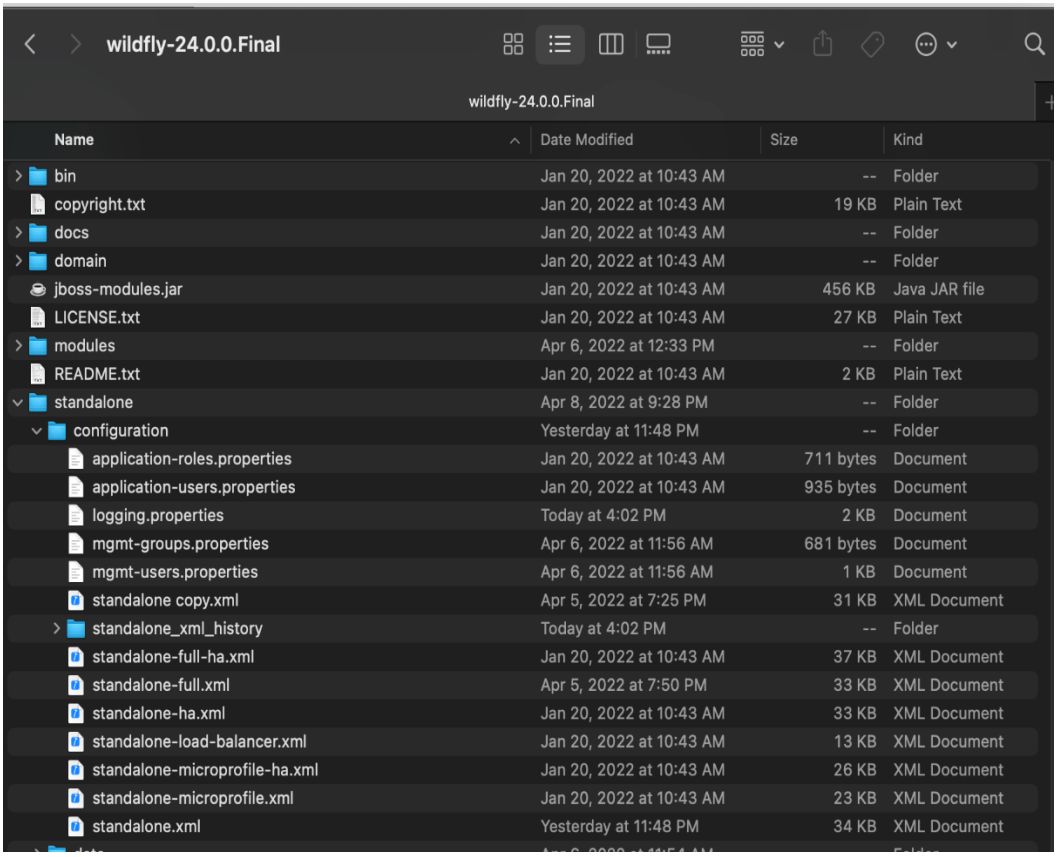
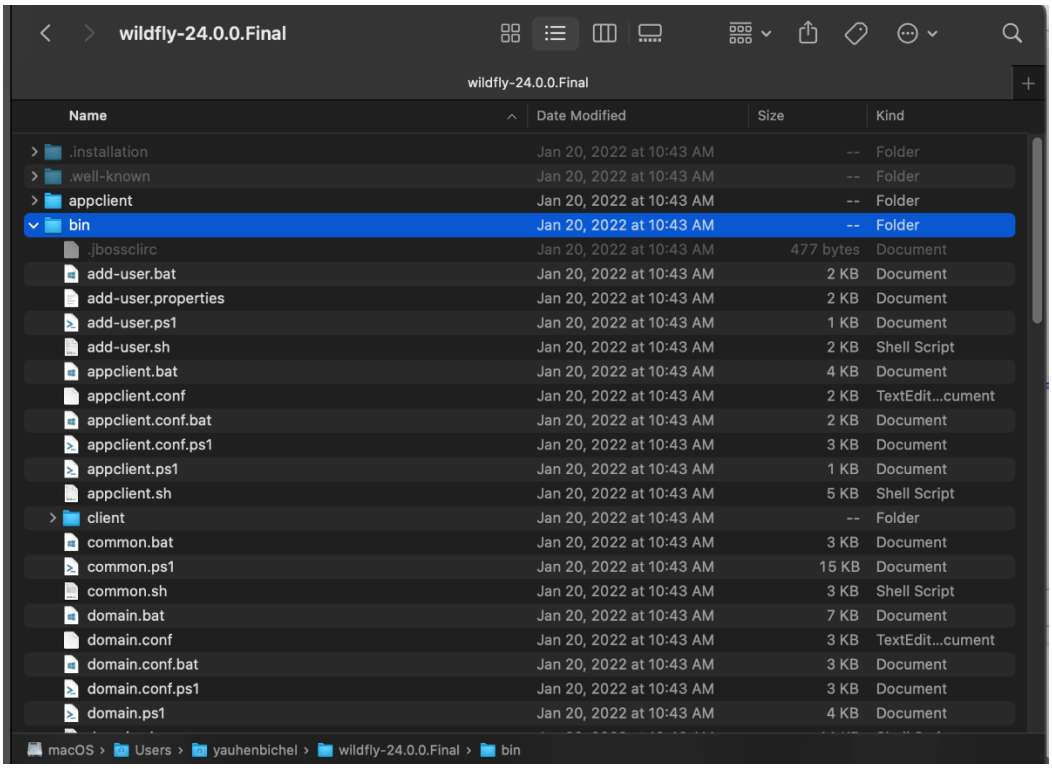


Figure 36 WildFly folder with standalone.xml file

   f. Copy standalone.xml file into wildfly-24.0.0.Final->standalone->configuration folder.



g. Setup WildFly in EclipseOpen Eclipse IDE
    i. Open Eclipse IDE
    ii. Click Window -> Show View -> Other -> Servers
    iii. Setup server by choosing WildFly 24+ and selecting folder the installed application server.
    iv. Add the application in the application server.
h. When the application server is running, open the management console http://127.0.0.1:9990/console/index.html
i. WildFly will ask to login. Then add a user using add-user.sh file in wildfly-24.0.0.Final -> bin folder. (comp1610y / password1)



j. Restart the application server and open the management console. After login, please check that datasourse is set (postgresql-42.3.3.jar), message contains topic and jms

k. Please find postgresql-42.3.3.jar jdbc driver in the application folder in src->main->wildfly folder.

l. Install all maven dependencies using command: >mvn install

m. Check the application is running by sending GET request http://127.0.0.1:9990/console/index.html

3. The application database is PostgreSQL, which is hosted in Heroku cloud provider. The database has the following settings:

a. Host: ec2-52-214-125-106.eu-west-1.compute.amazonaws.com

b. Database: dfejis77nibjdr

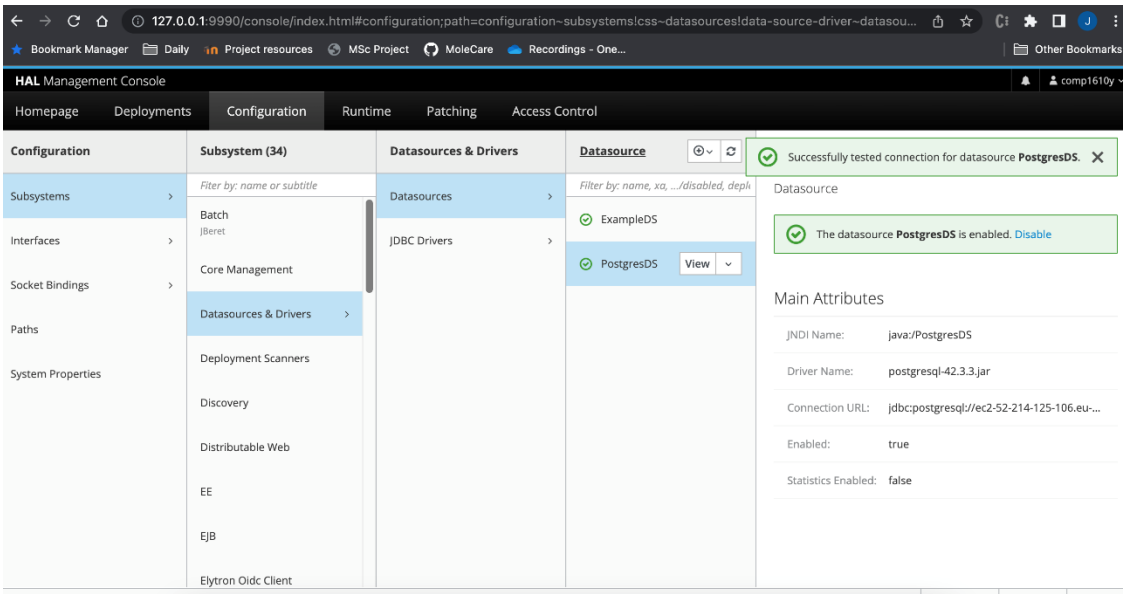
c. User: qexdvxvgdqtdgh

d. Port: 5432

e. Password: bf7f8df51b60c17d1fcc333c3f7b63e57df7300022deb67d60eb6cee2bd4e5dd

f. jdbc:postgresql://ec2-52-214-125-106.eu-west-1.compute.amazonaws.com:5432/dfejis77nibjdr

g. Heroku CLI: heroku pg:psql postgresql-infinite-73558 --app holiday-system



4. The mail box was created by me for the application in gmail with the following username and password: help.holiday.request@gmail.com / 1Vacation2!Request%

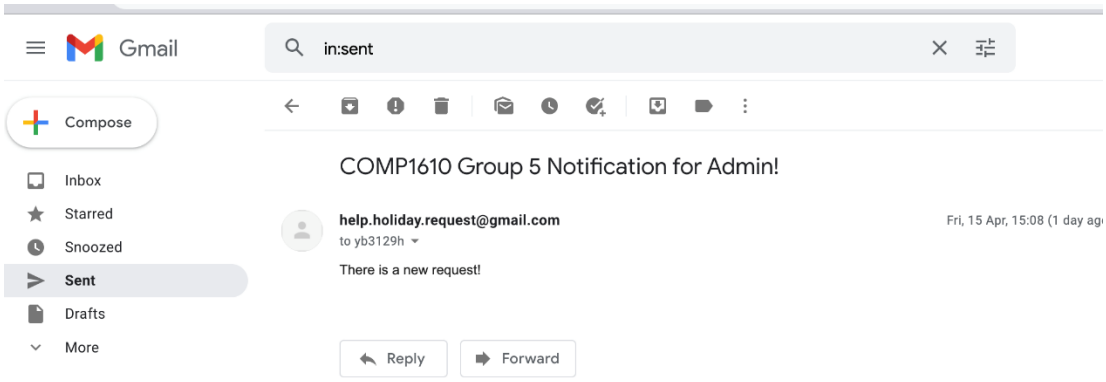


Figure 37 help.holiday.request@gmail.com mailbox

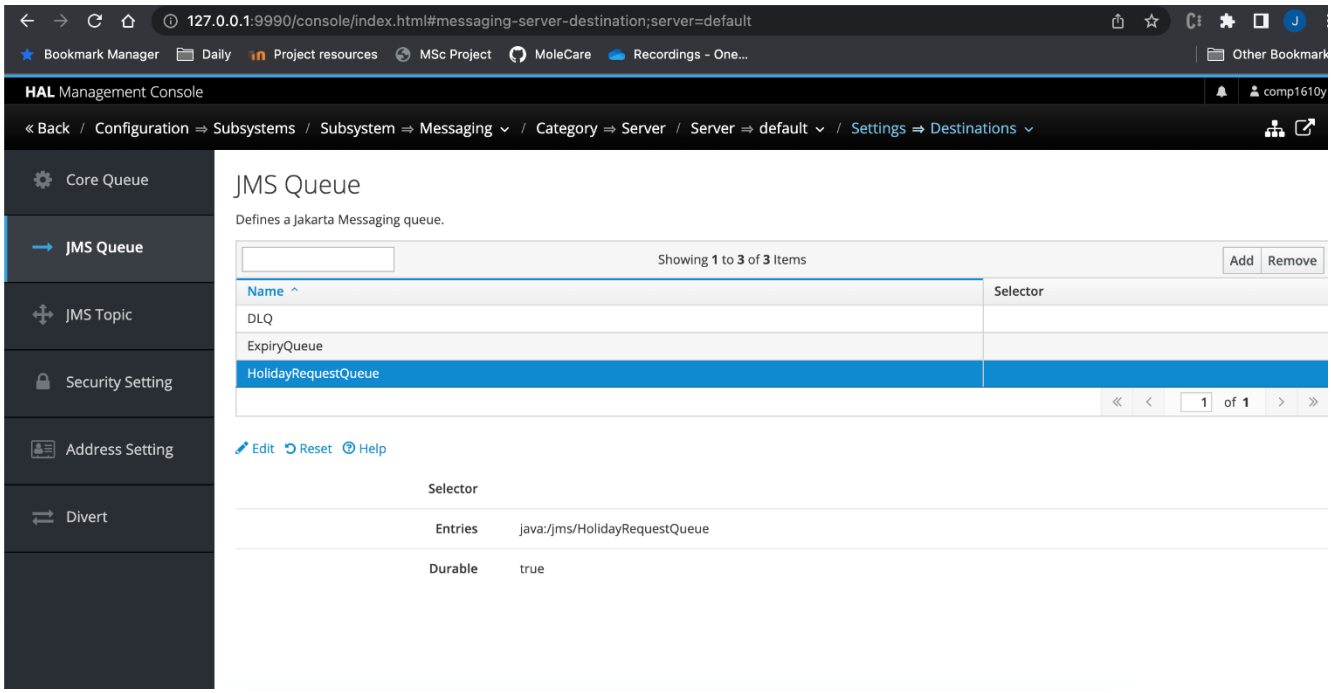


Figure 38 JMS Queue configuration

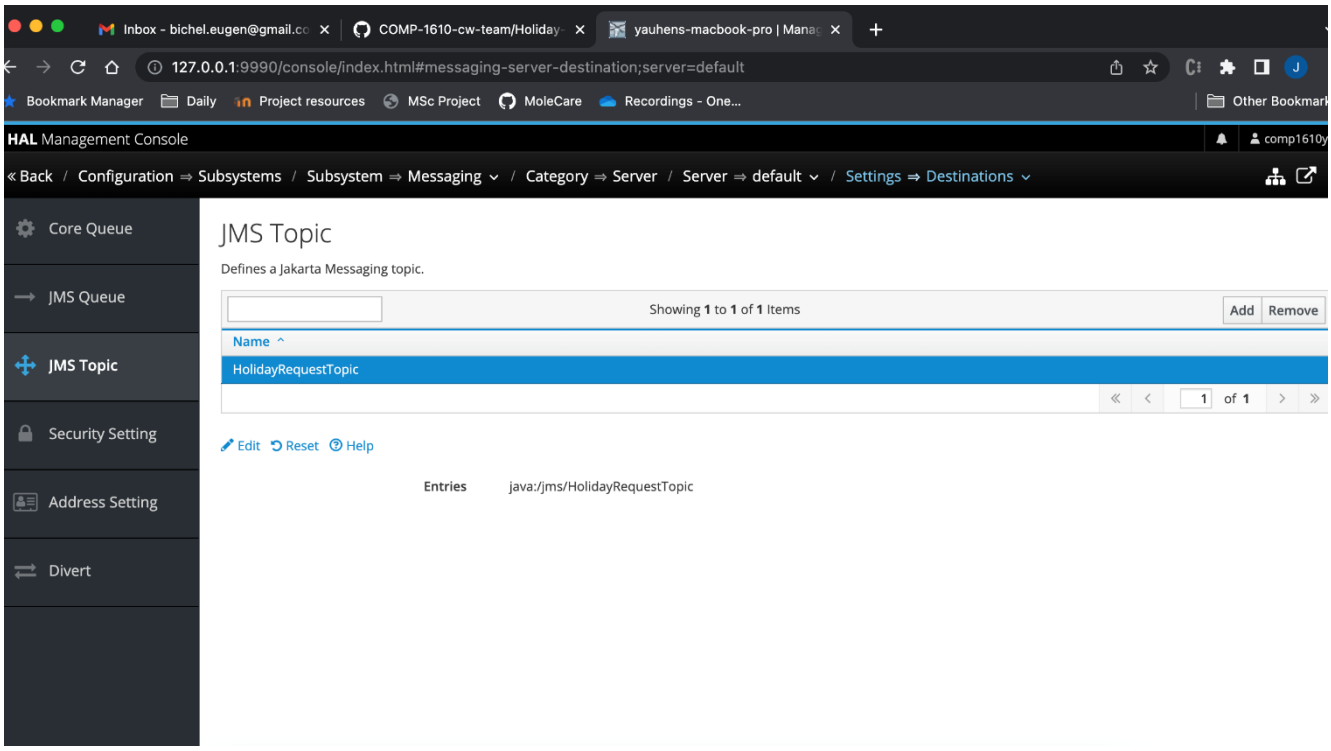


Figure 39 JMS Topic configuration

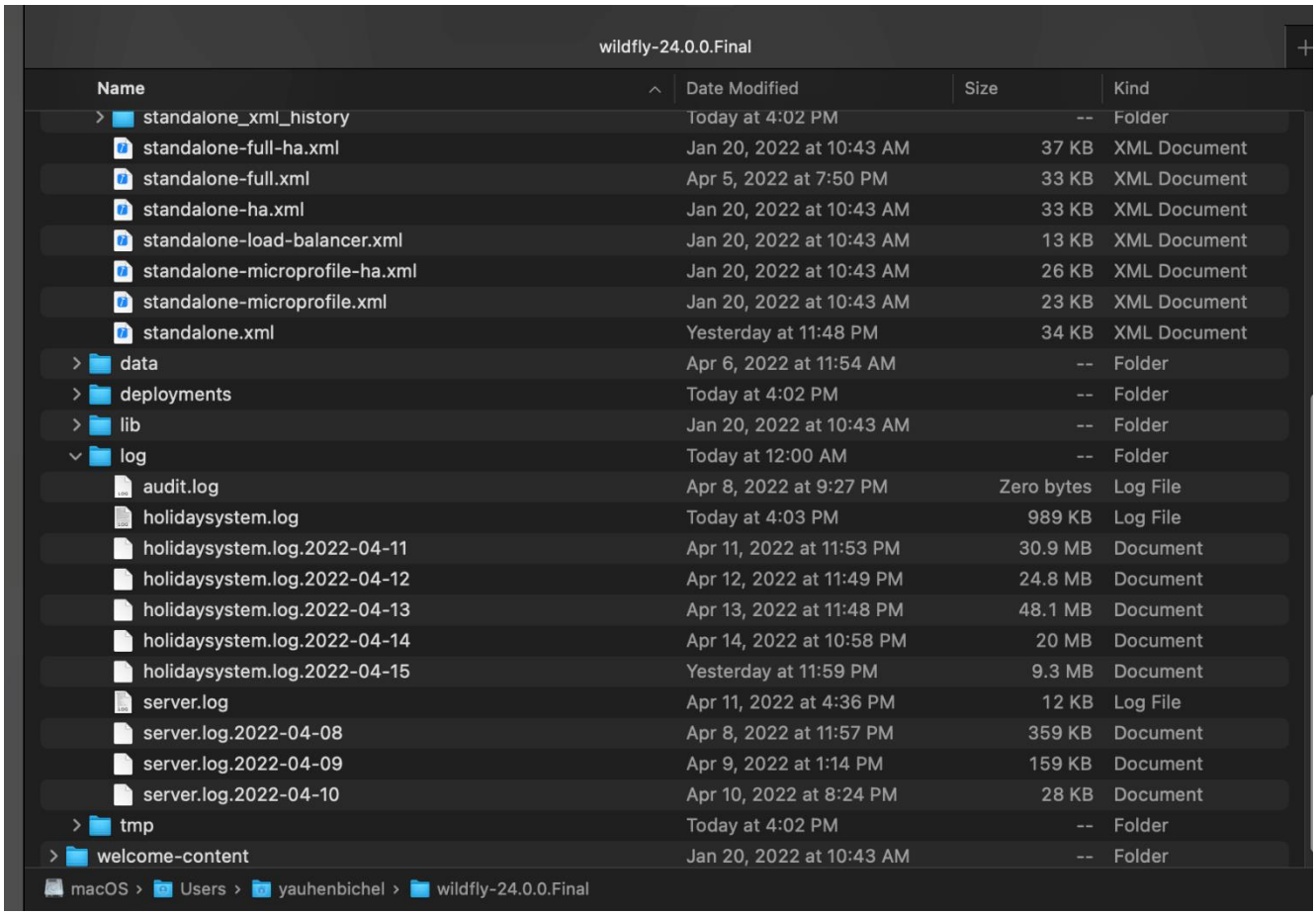
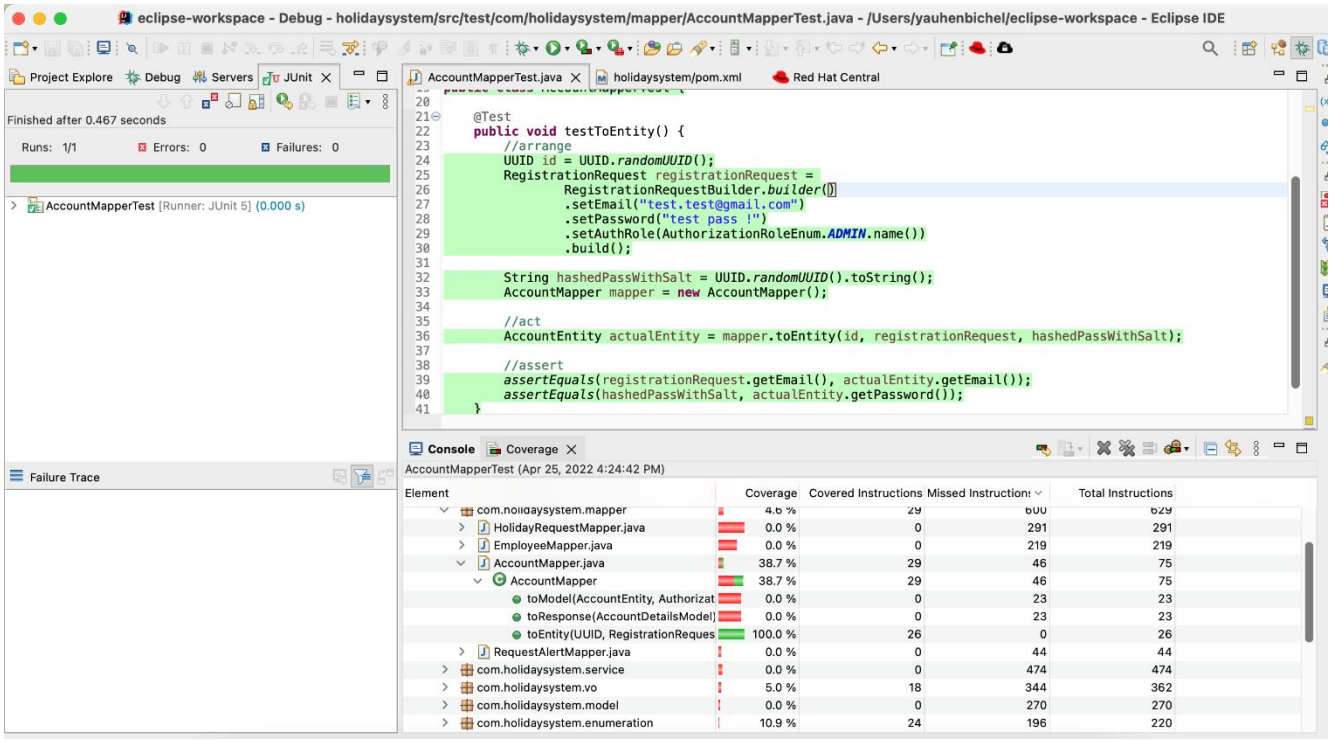


Figure 40 Logs are stored in log folder of WildFly

5. How to run unit tests

   Right mouse click and then run Junit tests



6. How to run desktop application

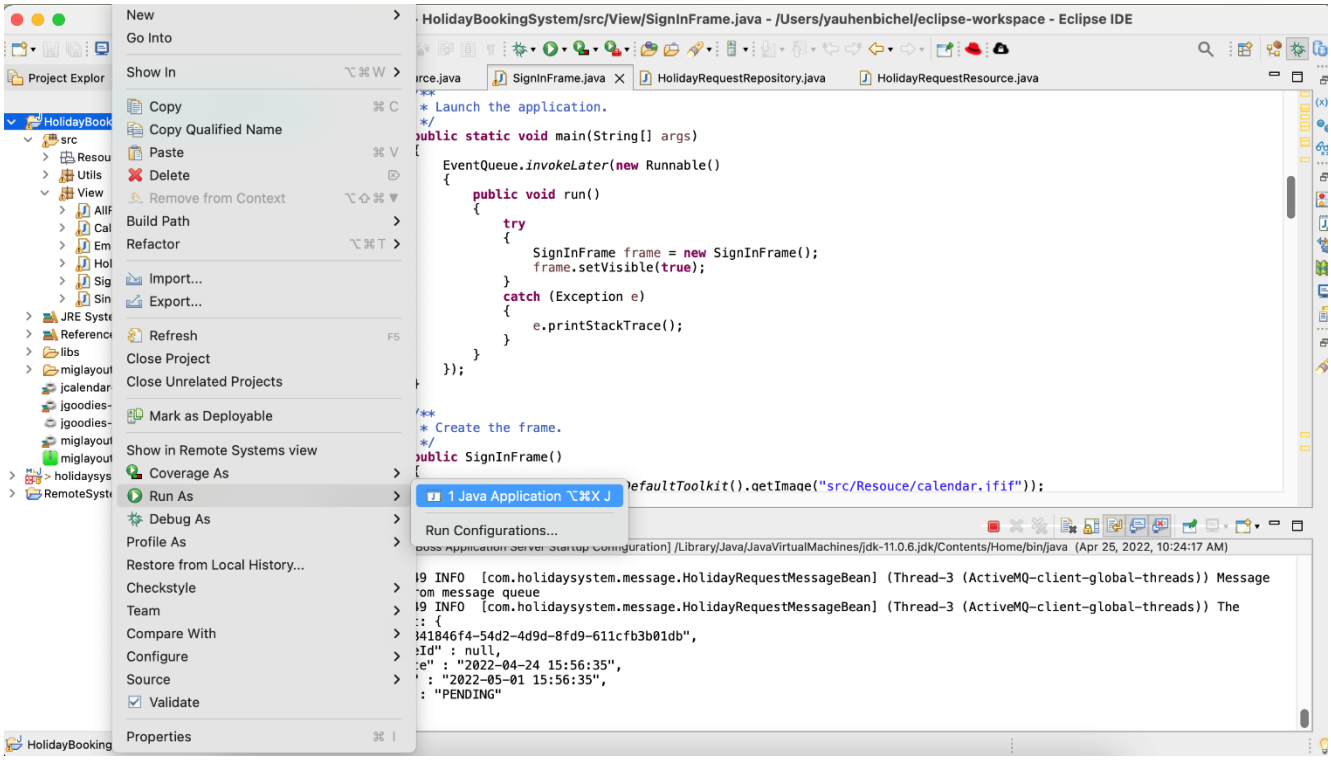


Figure 41 Right mouse click, choose Run As

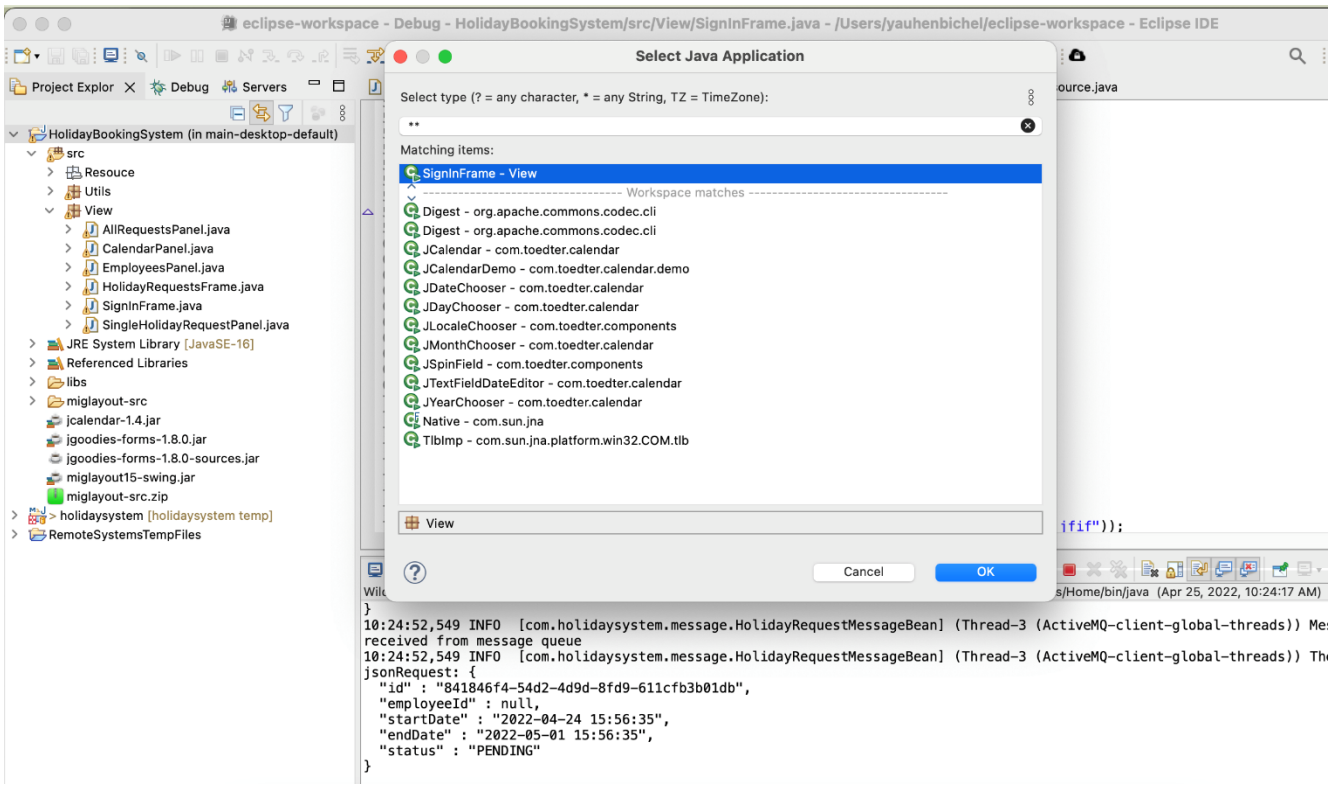


Figure 42 Choose SignIn frame

6. How to run web application

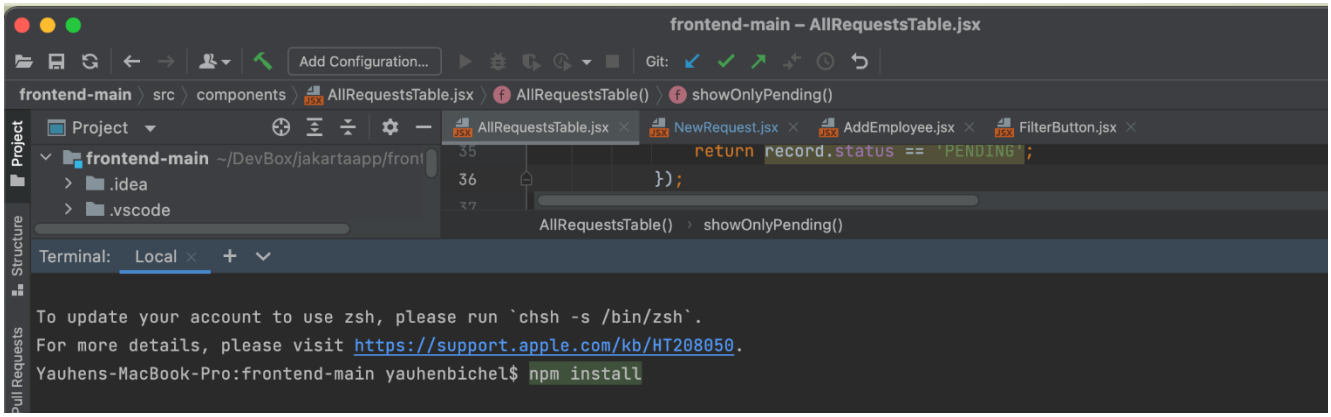


Figure 43 Download npm package dependencies

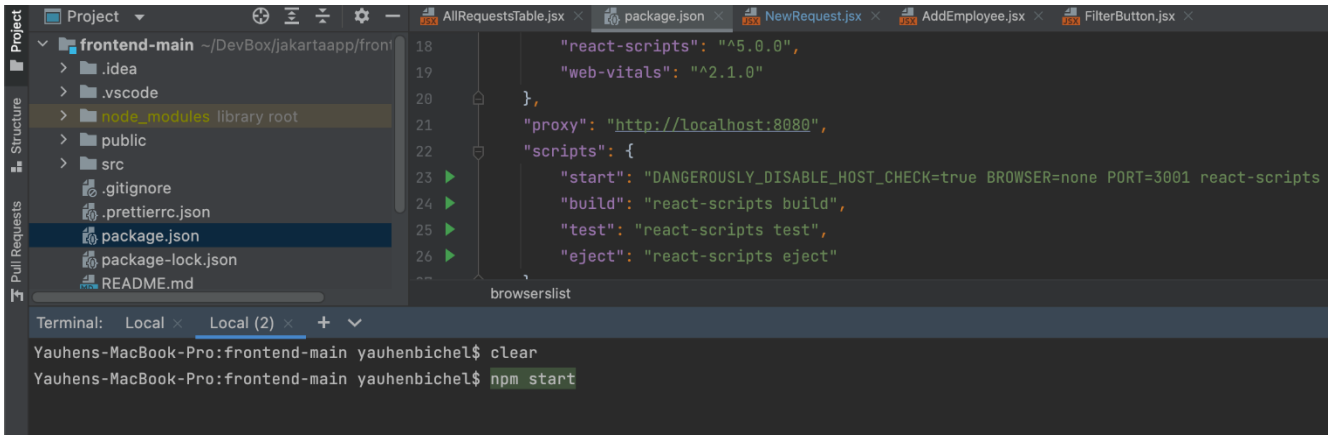


Figure 44 Run the following cmd to run the web app: npm start

7. Possible issues during starting wildfy application server
   If WildFly can not be started, then some ports can be taken by other applications. In this case, please find the processes Ids and kill them using the steps as provided in Figure 45.

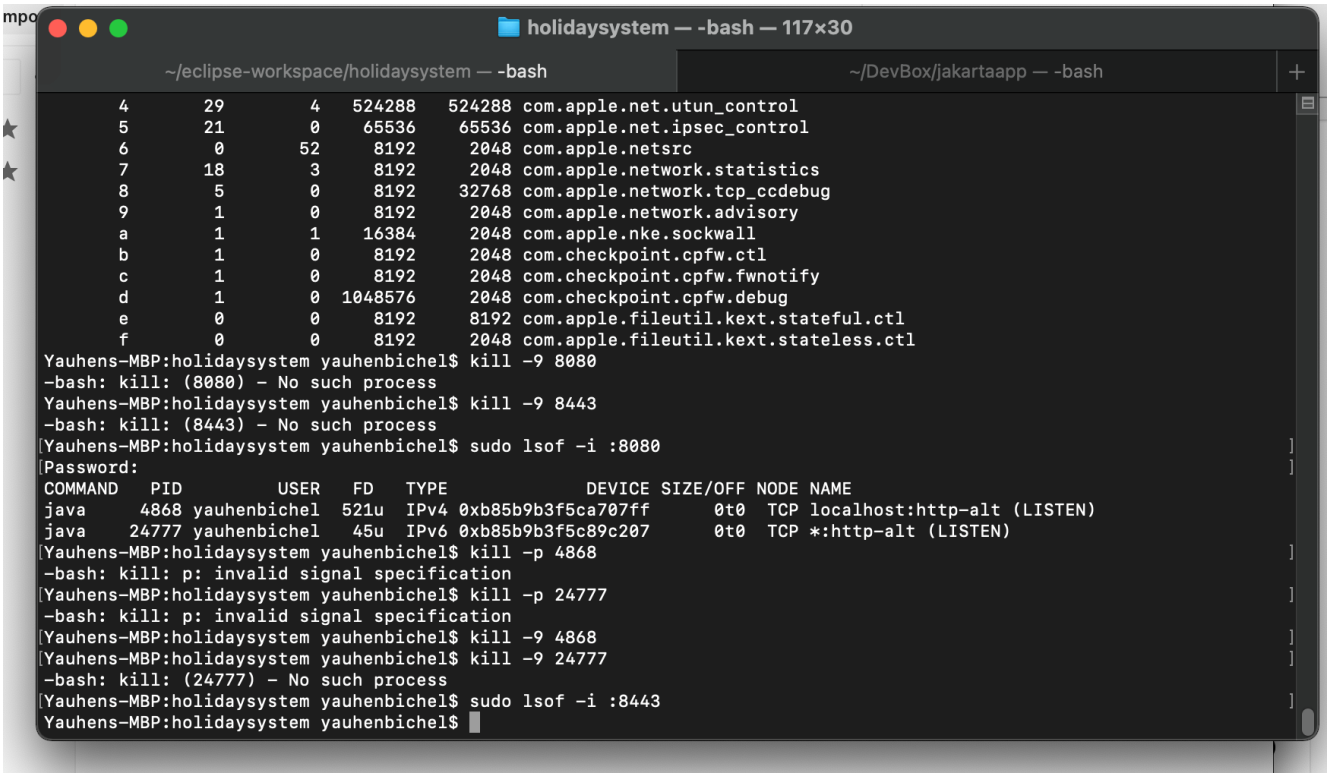


Figure 45 Find processes and kill them

8. REFERENCES

Docs.oracle.com, (2022), Standards and Specifications

[Online]

Available at:

https://docs.oracle.com/cd/E82085_01/140/rib_implementation_guide/Chapter%202%20-%20Standards%20and%20Specifications.htm

[Accessed 16 March 2022]

javaee.github.io/glassfish, (2022), GlassFish Documentation

[Online]

Available at: https://javaee.github.io/glassfish/documentation

[Accessed 12 March 2022]

9. BIBLIOGRAPHY

Martinez-Torres, Rafael, (2022), COMP 1610 Coursework Specification
[Online]
Available at: https://moodlecurrent.gre.ac.uk/mod/resource/view.php?id=1729371
[Accessed 10 Mar 2022]

Martinez-Torres, Rafael, (2022), Lectures, Programming Enterprise Components module materials
[Online]
Available at: https://moodlecurrent.gre.ac.uk/course/view.php?id=70195
[Accessed 05 January 2022]

Pressman, R., Maxim, B., (2019), Software Engineering: a practitioner's approach
[Online]
Available at: https://ebookcentral.proquest.com/lib/gre/reader.action?docID=5989441
[Accessed 25 Feb 2022]

Eclipse-ee4j.github.io, (2022), The Jakarta EE Tutorial
[Online]
Available at: https://eclipse-ee4j.github.io/jakartaee-tutorial/#introduction-to-jakarta-ee
[Accessed 14 March 2022]

Devcenter.heroku.com, (2022), Heroku Postgres
[Online]
Available at: https://devcenter.heroku.com/articles/heroku-postgresql
[Accessed 10 January 2022]

Baeldung.com, (2022), How to Set Up a WildFly Server
[Online]
Available at: https://www.baeldung.com/wildfly-server-setup
[Accessed 01 February 2022]

Wildfly.org, (2022), WildFly Documentation
[Online]

Available at: https://docs.wildfly.org/26/#administrator-guides
[Accessed 01 March 2022]

Eclipse-ee4j.github.io, (2022), The Jakarta EE Tutorial
[Online]
Available at: https://eclipse-ee4j.github.io/jakartaee-tutorial/#introduction-to-jakarta-ee
[Accessed 14 March 2022]

Jakarta.ee, (2022), Specifications
[Online]
Available at: https://jakarta.ee/specifications/
[Accessed 12 March 2022]

Docs.oracle.com, (2022), Oracle WebLogic Server
[Online]
Available at: https://docs.oracle.com/en/middleware/fusion-middleware/weblogic-server/
[Accessed 12 March 2022]