





Lab 8-12 Review

Lab-08 Skeleton 




Develop a baseline for Assignment 2, to include a simplified version of pacemaker application developed so far


Lab-09 Rest API 




REST API


Evolve a simple Rest service from the existing pacemaker-skeleton app using the Javalin microframework.

Lab-10 Rest CLI 





Implement a new project that will be a client of the pacemaker-skeleton application.

Lab-11 Rest Test 

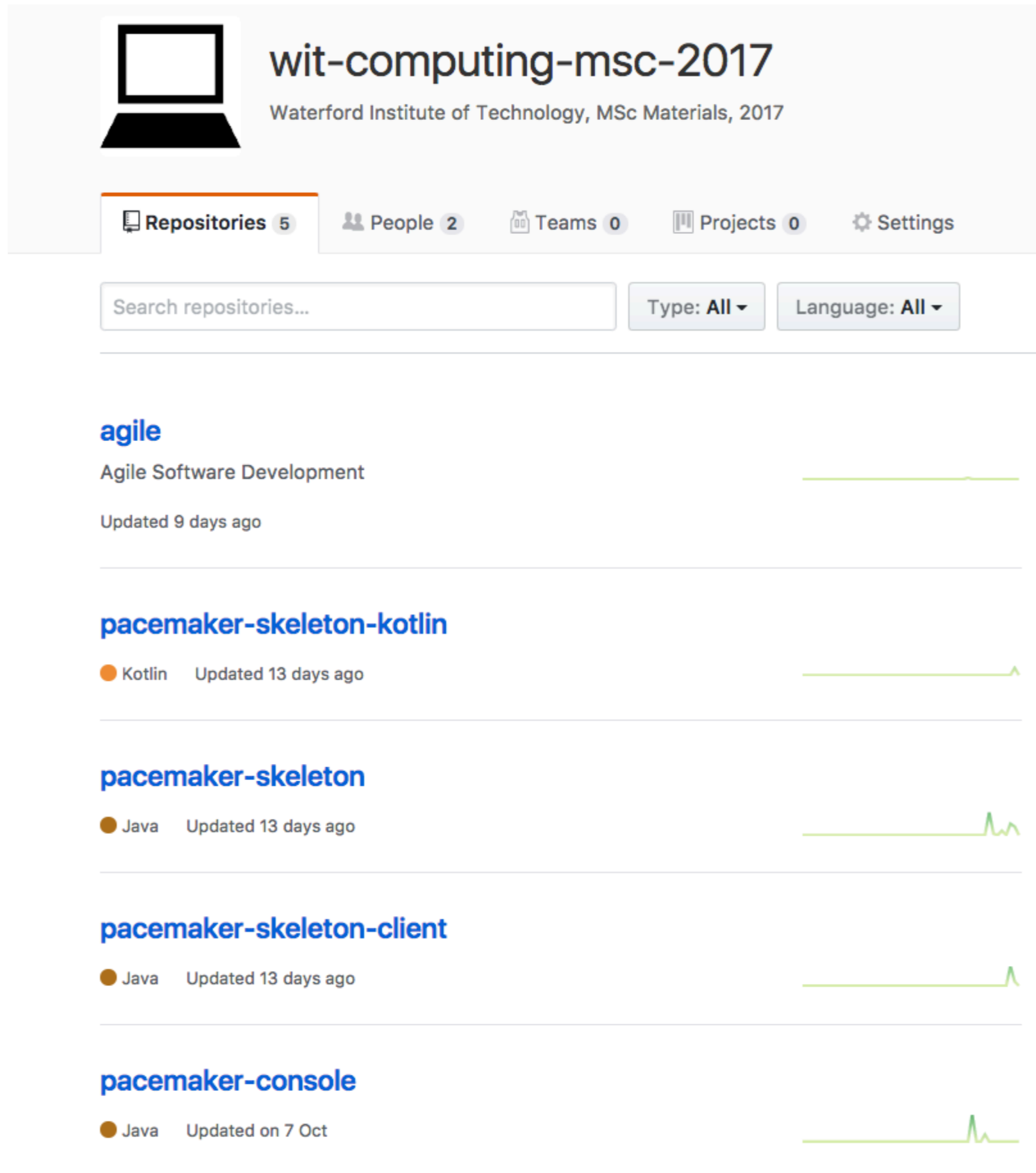


Complete the API + write a range of unit tests to exercises the features. Deploy the service to the cloud.

Lab-12 Kotlin 



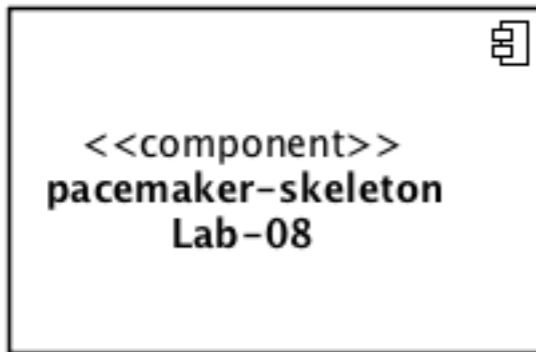
Introduce a new project, an implementation of the service in Kotlin.



The image shows a GitHub repository page for 'wit-computing-msc-2017'. At the top left is a laptop icon. To its right is the repository name 'wit-computing-msc-2017' and the subtitle 'Waterford Institute of Technology, MSc Materials, 2017'. Below this is a navigation bar with 'Repositories 5' (highlighted), 'People 2', 'Teams 0', 'Projects 0', and 'Settings'. A search bar contains 'Search repositories...'. To its right are filters for 'Type: All' and 'Language: All'. The repository list includes:

- agile**: Agile Software Development, Updated 9 days ago. A green line graph shows activity.
- pacemaker-skeleton-kotlin**: Kotlin, Updated 13 days ago. A green line graph shows activity.
- pacemaker-skeleton**: Java, Updated 13 days ago. A green line graph shows activity.
- pacemaker-skeleton-client**: Java, Updated 13 days ago. A green line graph shows activity.
- pacemaker-console**: Java, Updated on 7 Oct. A green line graph shows activity.

Lab 08



Standalone
Java Console
Application



Console UX



Lab 08 pacemaker-skeleton

<https://github.com/wit-computing-msc-2017/pacemaker-skeleton/releases/tag/lab08.exercises>



Develop a baseline for Assignment 2, to include a simplified version of pacemaker application developed so far

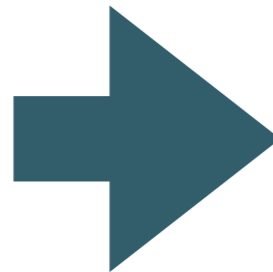
```

pacemaker-skeleton [pacemaker-skeleton]
├── src/main/java
│   ├── controllers
│   │   ├── Main.java
│   │   ├── PacemakerAPI.java
│   │   └── PacemakerConsoleService.java
│   ├── models
│   │   ├── Activity.java
│   │   ├── Location.java
│   │   └── User.java
│   └── parsers
│       ├── AsciiTableParser.java
│       └── Parser.java
    
```



- lab09.start dependencies for Javalin
- lab08.exercises add location, list locations + activity report
- lab08.end placeholder for tests
- add and list activity commands
- implement register, list users, login and logout commands
- console service - stubbed implementation of all commands
- PacemakerAPI simplified to exclude serialisation
- parsers for asciitable
- simplified models - start time and duration removed
- initial pom + gitignore

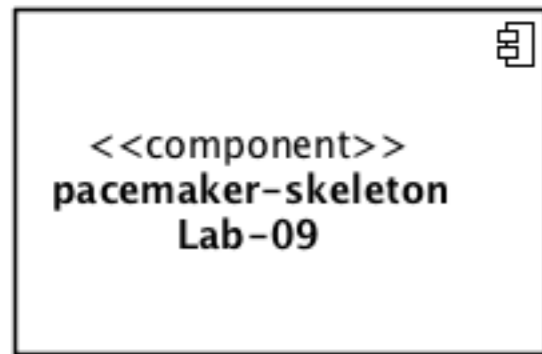
Implements
Baseline
commands



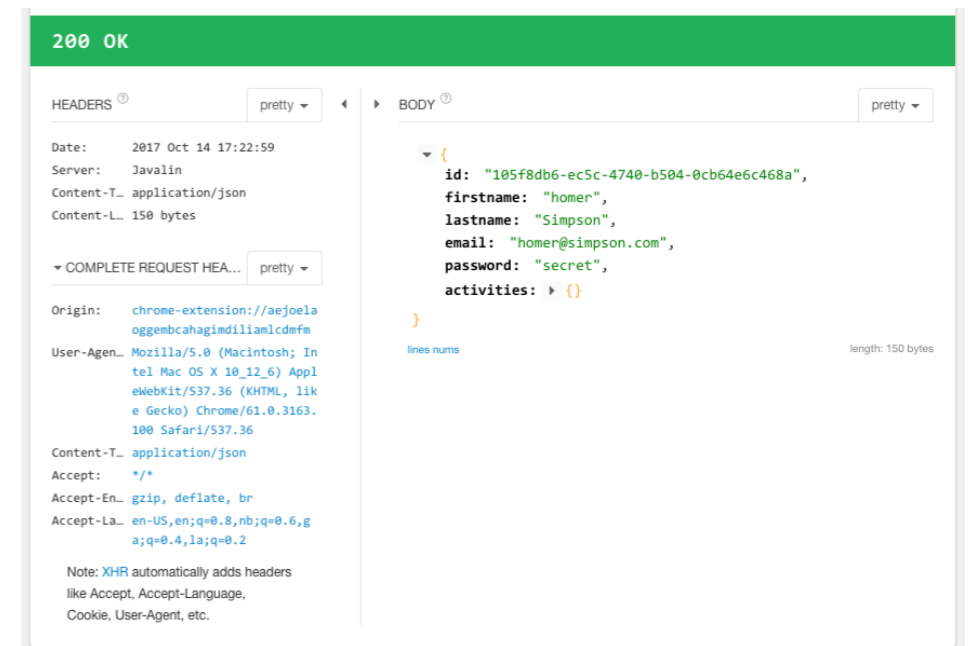
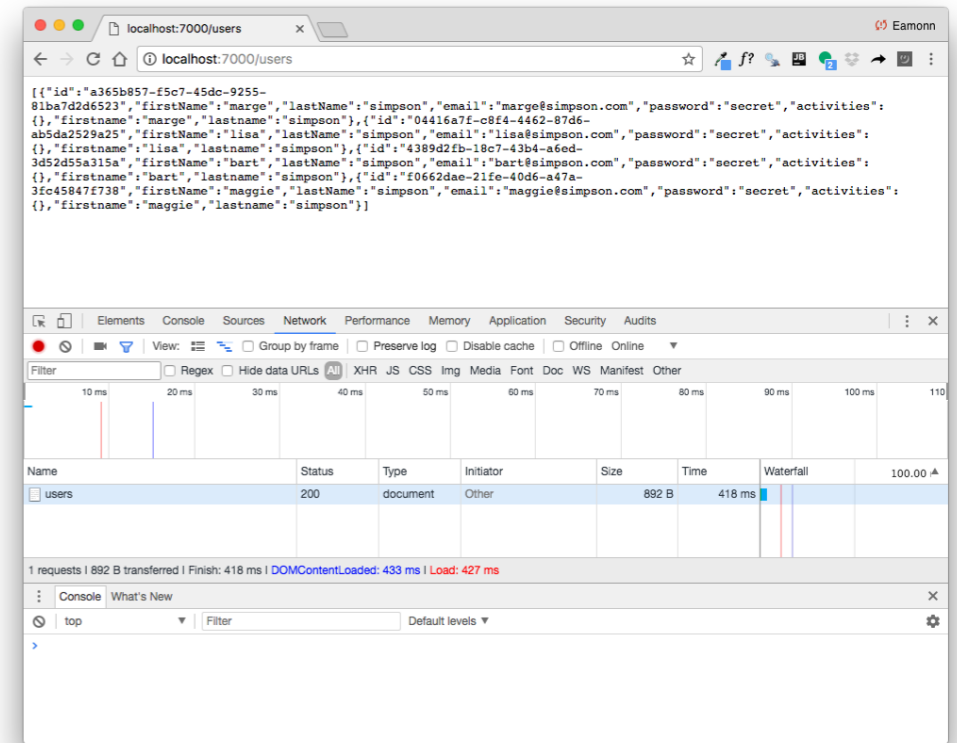
Grade Band	Packaging & Deployment	Commands	TDD Coverage	Language
Starter	Eclipse project archive - pacemaker-console	gu get-users () ru register-user (first name, last name, email, password) lu login-user (email, password) l logout () aa add-activity (type, location,	30%	Java
Baseline	github repo - pacemaker-console	al add-location (activity-id, longitude, latitude) lal list-activity-locations (activity-id) ar activity-report () f follow (email) lf list-friends () far friend-activity-report (email)	40%	Java
Good	maven github repos: - pacemaker-service - pacemaker-console	ar activity-report (byType: type) uf unfollow-friend () mf message-friend (email, message) lm list-messages () dlb distance-leader-board ()	50%	Java with Lambdas

Lab 09

Browser



Web Service
Implementing
REST
Endpoints



REST Client
(Restlet)



Lab 09

<https://github.com/wit-computing-msc-2017/pacemaker-skeleton/releases/tag/lab09.exercises>



Evolve a simple Rest service from the existing pacemaker-skeleton app using the Javalin microframework.

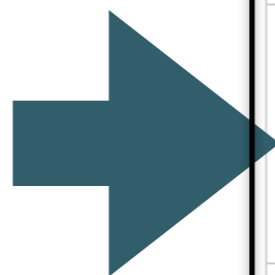
```

pacemaker-skeleton [pacemaker-skeleton r
└─ src/main/java
  └─ controllers
    ├── PacemakerAPI.java
    ├── PacemakerRestService.java
    └─ RestMain.java
  └─ models
    ├── Activity.java
    ├── Fixtures.java
    ├── Location.java
    └─ User.java
    
```



- master 13 behind lab10.start removed cli dependent classes
- lab09.exercises getActivity + activity location routes
- lab09.end create activity and list activities routes support request for individual user based in id complete plugin update correct activity report implementations remove camelcase from attribute names preload test users and return user endpoint use fixture containing test users first version of rest service + /users route implementations new main to launch web service
- lab09.start dependencies for Javalin

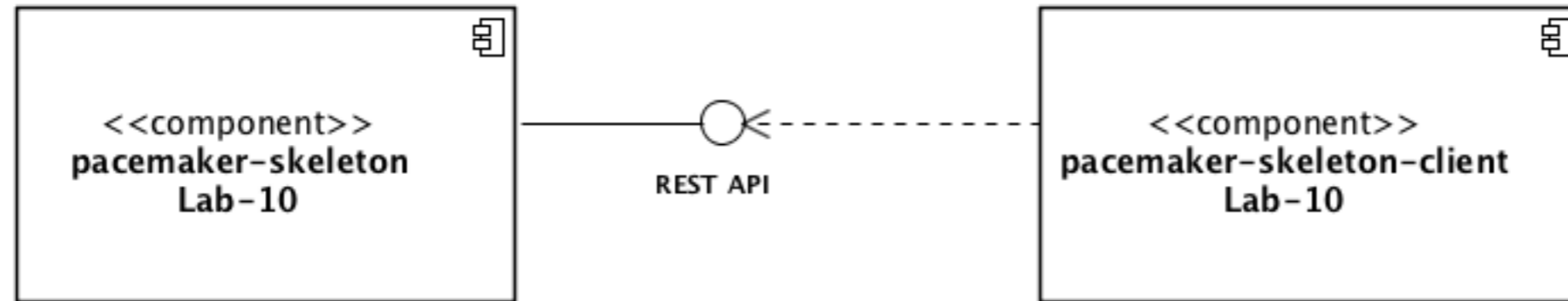
Implements REST Service



Good	maven github repos: - pacemaker-service - pacemaker-console	<pre> far friend-activity-report (email) ar activity-report (byType: type) uf unfollow-friend () mf message-friend (email, message) lm list-messages () qlb distance-leader-board () </pre>	50%	Java with Lambdas
Excellent	pacemaker-service provides REST API pacemaker-console access API (over http)	<pre> dlb distance-leader-board-by-type (byType: type) maf message-all-friends (message) llb location-leader-board (location) </pre>	65%	Java with Streams OR Kotlin
Outstanding	pacemaker-service deployed to cloud pacemaker-client access cloud service	<pre> Admin Account Define commands to administer service, to include: - remove users - disable/enable users - report user stats (nmr logins, average number of activities etc...) </pre>	80% With Mocking	Kotlin

Lab 10

Web Service Implementing REST Endpoints



+ additional
endpoints

Console UX

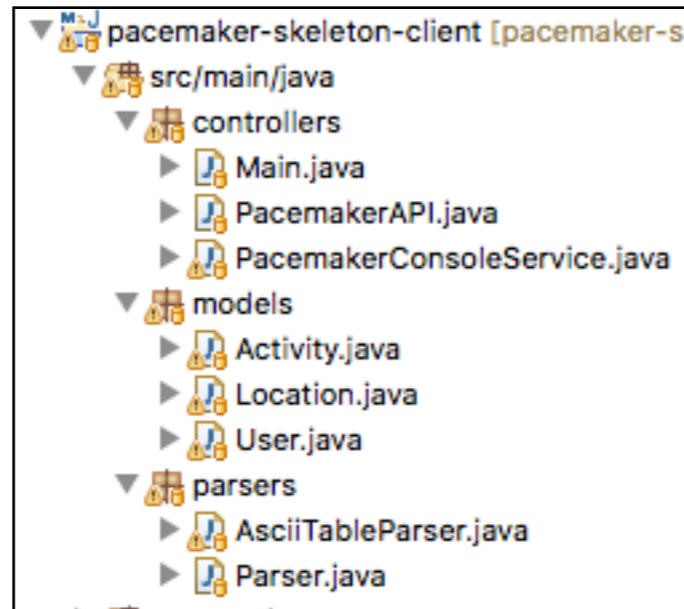


Lab-10 Rest CLI



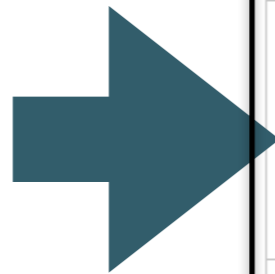
Implement a new project that will be a client of the pacemaker-skeleton application.

<https://github.com/wit-computing-msc-2017/pacemaker-skeleton-client/releases/tag/lab10.exercises>



- lab10.exercises addLocations exercise implementation
- lab10.end simplified client user models - removed id generation and one-to-one relationships
- remove fixtures
- get and add activity commands
- remote api incorporated in PacemakerAPI
- initial remote api interface incorporated into PacemakerAPI
- retrofit libraries
- replace PacemakerAPI with stubbed implementation
- remove rest libraries
- remove rest features
- lab10.start cloned from pacemaker-skeleton project

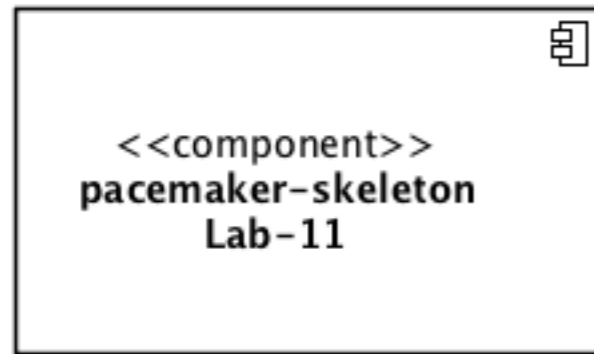
Implements REST Client



Good	maven github repos: - pacemaker-service - pacemaker-console	<pre> far friend-activity-report (email) ar activity-report (byType: type) uf unfollow-friend () mf message-friend (email, message) lm list-messages () qlb distance-leader-board () </pre>	50%	Java with Lambdas
Excellent	pacemaker-service provides REST API pacemaker-console access API (over http)	<pre> dlb distance-leader-board-by-type (byType: type) maf message-all-friends (message) llb location-leader-board (location) </pre>	65%	Java with Streams OR Kotlin
Outstanding	pacemaker-service deployed to cloud pacemaker-client access cloud service	<pre> Admin Account Define commands to administer service, to include: - remove users - disable/enable users - report user stats (nmr logins, average number of activities etc...) </pre>	80% With Mocking	Kotlin

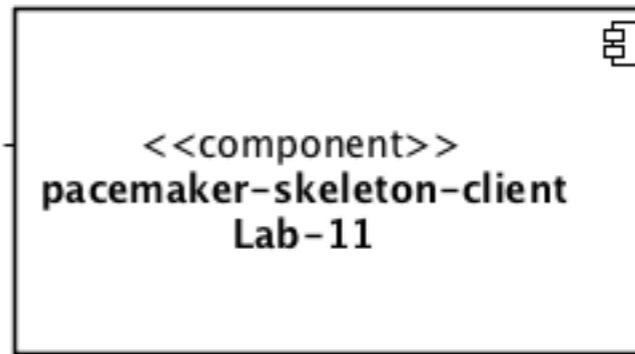
Lab 11

Web Service
Implementing
REST
Endpoints

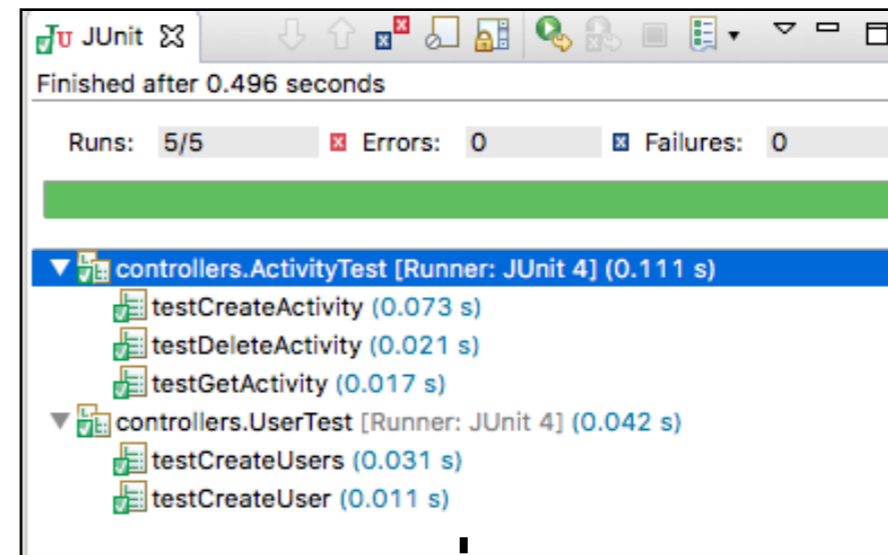


+ additional endpoints
+ cloud deployment (heroku)

REST API



Console UX



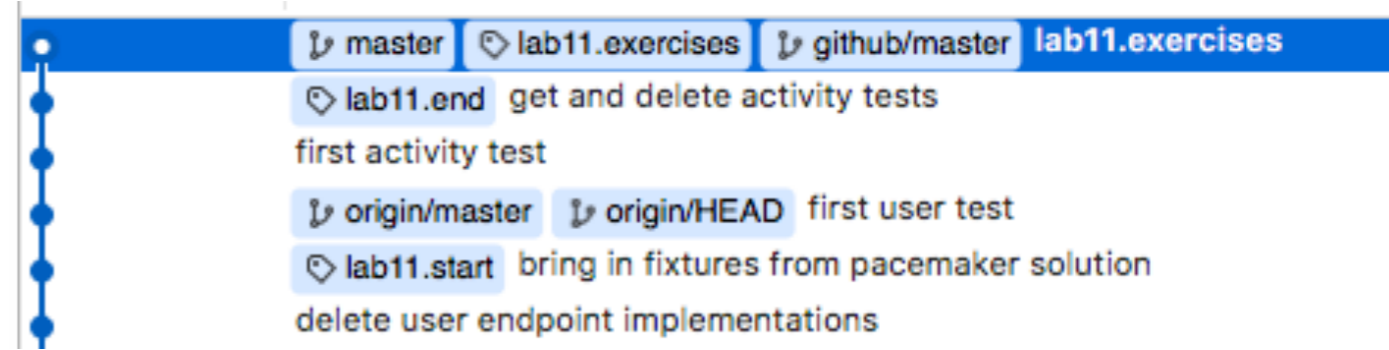
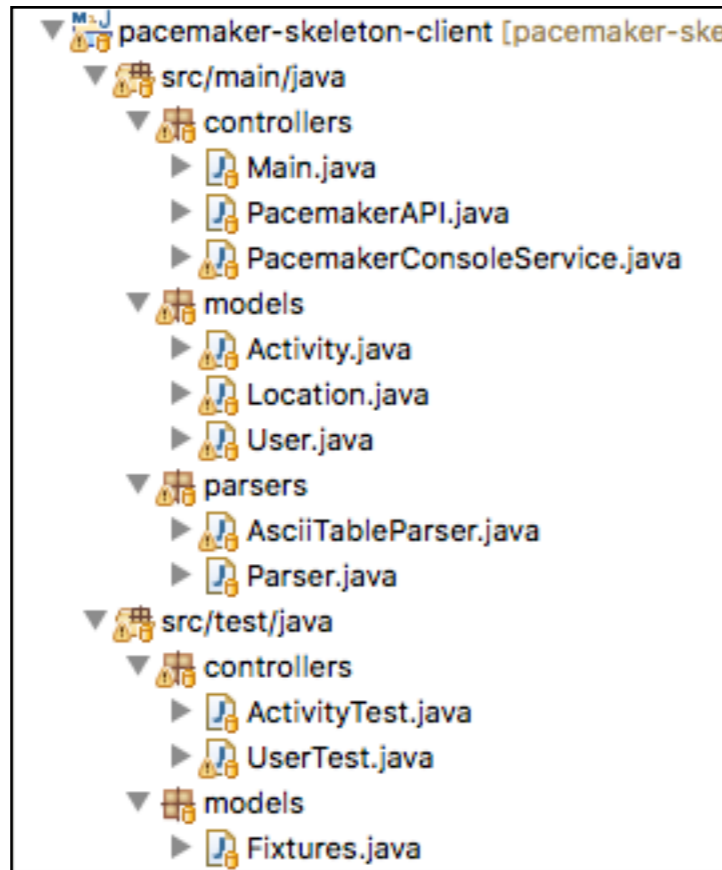
JUnit
Tests

Lab-11 Rest Test



Complete the API + write a range of unit tests to exercises the features. Deploy the service to the cloud.

<https://github.com/wit-computing-msc-2017/pacemaker-skeleton-client/releases/tag/lab11.exercises>



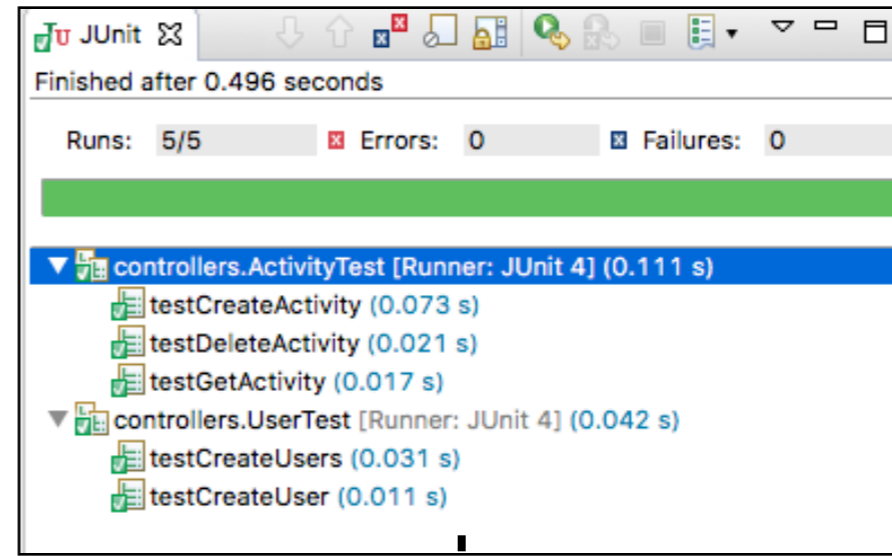
Implements REST Client Unit Tests

Deploy REST Service to Cloud

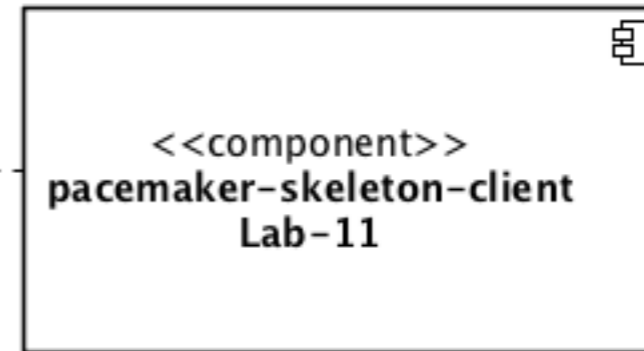
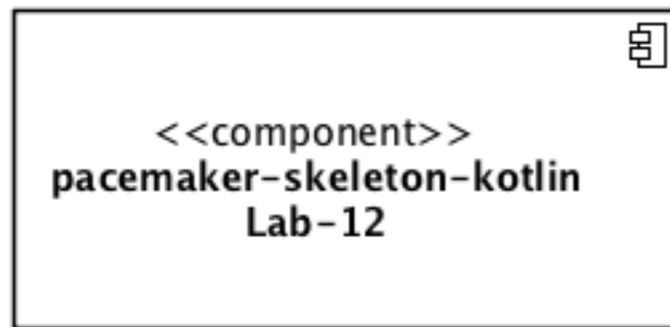


Good	maven github repos: - pacemaker-service - pacemaker-console	far friend-activity-report (email) ar activity-report (byType: type) uf unfollow-friend () mf message-friend (email, message) lm list-messages () dlb distance-leader-board ()	50%	Java with Lambdas
Excellent	pacemaker-service provides REST API pacemaker-console access API (over http)	dlbdt distance-leader-board-by-type (byType: type) maf message-all-friends (message) llb location-leader-board (location)	65%	Java with Streams OR Kotlin
Outstanding	pacemaker-service deployed to cloud pacemaker-client access cloud service	Admin Account Define commands to administer service, to include: - remove users - disable/enable users - report user stats (nmr logins, average number of activities etc...)	80% With Mocking	Kotlin

Lab 12



JUnit Tests



Kotlin Version
of Web Service
Implementing
REST
Endpoints

Console UX



Lab-12 Kotlin



<https://github.com/wit-computing-msc-2017/pacemaker-skeleton-kotlin>



Introduce a new project, an implementation of the service in Kotlin.

```
pacemaker-skeleton-kotlin
├── src/main/kotlin
│   ├── controllers
│   │   ├── PacemakerAPI.kt
│   │   ├── PacemakerRestService.kt
│   │   └── RestMain.kt
│   └── models
│       ├── Activity.kt
│       ├── Location.kt
│       └── User.kt
```

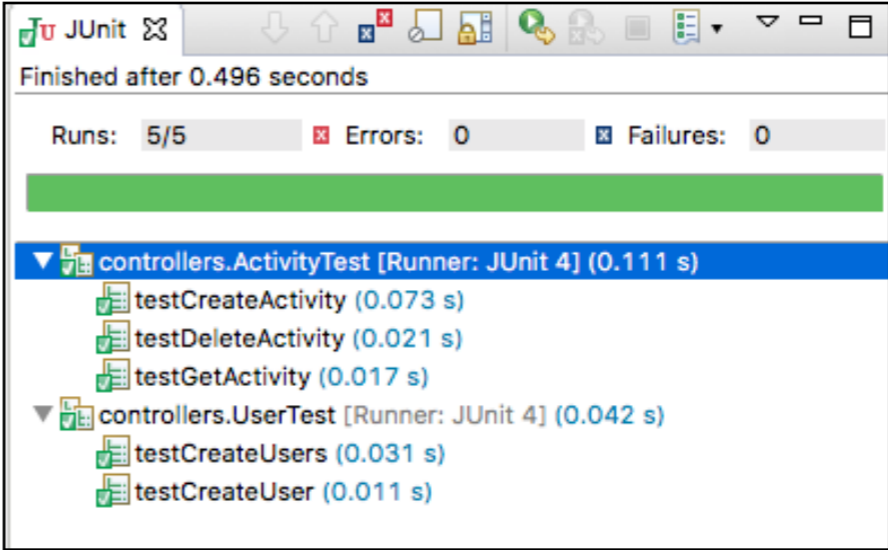
```

┆ master
┆ lab12.end
┆ origin/master initial activity endpoints
expose user endpoints + main program
core model implementation in PacemakerAPI
Location, Activity and User models
┆ lab12.start starter kotlin project - with javalin + heroku support
    
```

		<pre> far friend-activity-report (email) ar activity-report (byType: type) uf unfollow-friend () mf message-friend (email, message) lm list-messages () dlb distance-leader-board () </pre>		
Good	maven github repos: - pacemaker-service - pacemaker-console		50%	Java with Lambdas
Excellent	pacemaker-service provides REST API pacemaker-console access API (over http)	<pre> dlbdt distance-leader-board-by-type (byType: type) maf message-all-friends (message) llb location-leader-board (location) </pre>	65%	Java with Streams OR Kotlin
Outstanding	pacemaker-service deployed to cloud pacemaker-client access cloud service	Admin Account Define commands to administer service, to include: - remove users - disable/enable users - report user stats (nmr logins, average number of activities etc...)	80% With Mocking	Kotlin

← Kotlin version of REST Service

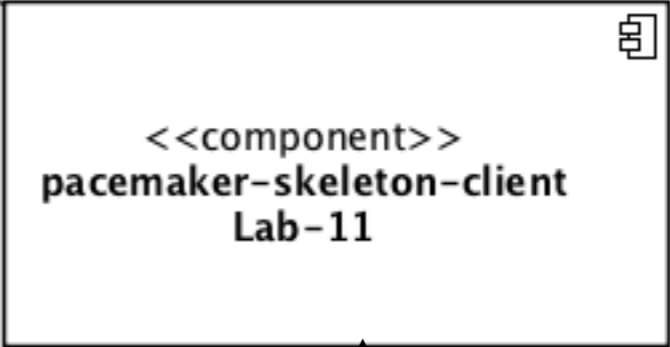
Summary



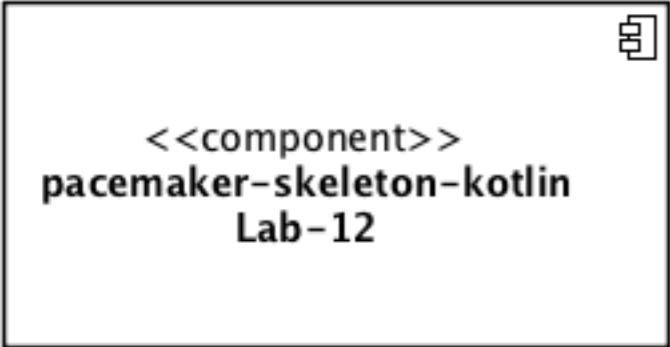
JUnit Tests



REST API



Console UX



REST API

