Team Badger

Programming Languages

Final report

4-30-2020

To Do Kotlin app

General Information

- Title of the Project
  - To-Do list app
- Your Team Name and team members' names
  - Team Badger
  - Peter Keres, Kevin Horadan, Nicholas White.

Introduction

- Goal
  - To gain a better understanding of what the Kotlin programming language can do. This includes acts such as language syntax, technology connectivity, and library support.
- What did you try to do?
  - Make a ‘To-do’ list application. This application would keep a track of any To-do items a user wanted to keep track of. This application would have a GUI, this would allow us to test some graphics library attached to Kotlin. The graphics library we used is called ‘Tornado FX’. We also want to host the database for this project on AWS. For this, we used a combination of Hibernate and PostgreSQL DBMS.
- User Stories

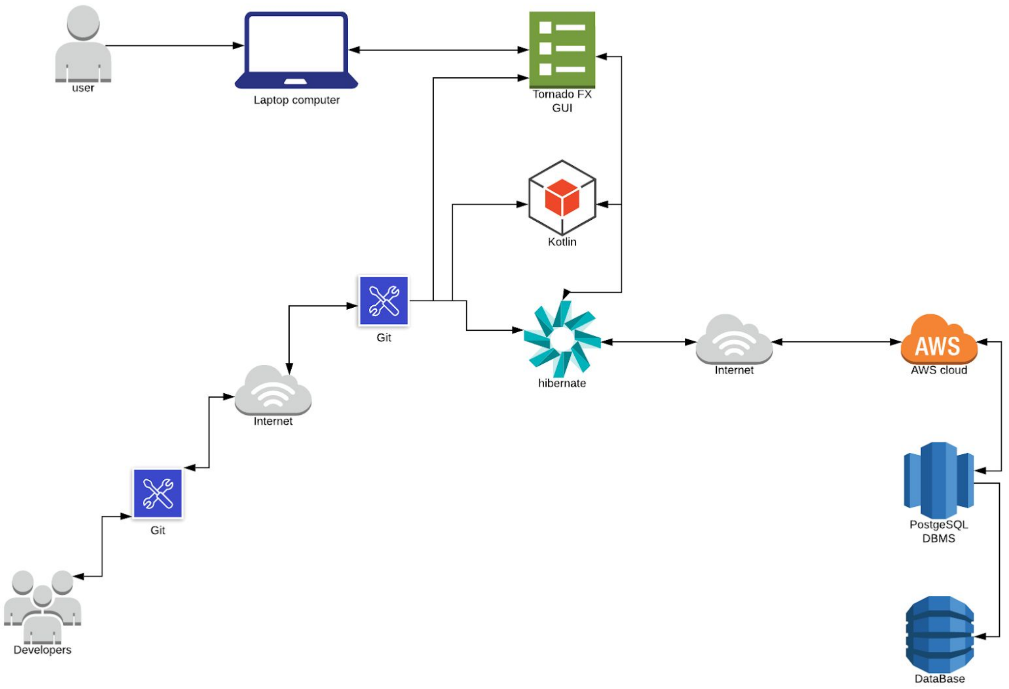
- A user enter tasks
    - A user will type in some type of task into the system. The system will then save this task somewhere so when the user checks back on it, it's still there.
    - medium
    - High
- A user can view tasks
    - A user should be able to view the tasks in the system. When viewing tasks, they should see all aspects of the task they created.
    - short
    - High
- A user can delete tasks
    - A user should be able to delete any one of their tasks from the system. They should allow deleting multiple tasks at once. Also, should be prompted on delete if it is ok.
    - Medium
    - High
- A user has to log into system
    - A user needs to log into the system before they are allowed to do any actions on the system.
    - Long
    - Normal
- Different users on the system
    - There are different users in the system. Each user has its own set of tasks that the system tracks. Need to allow the creation of new users
    - Long
    - Low
- A user can sort their tasks
    - When a user looks at their tasks, they should be able to view them in a few different orders. By name, data made, etc.
    - Medium
    - Normal
- User can tag their tasks by date or priority
    - The user should be able to add tags to each task, such as due date or priority
    - How long to develop: normal

        - How important: mid

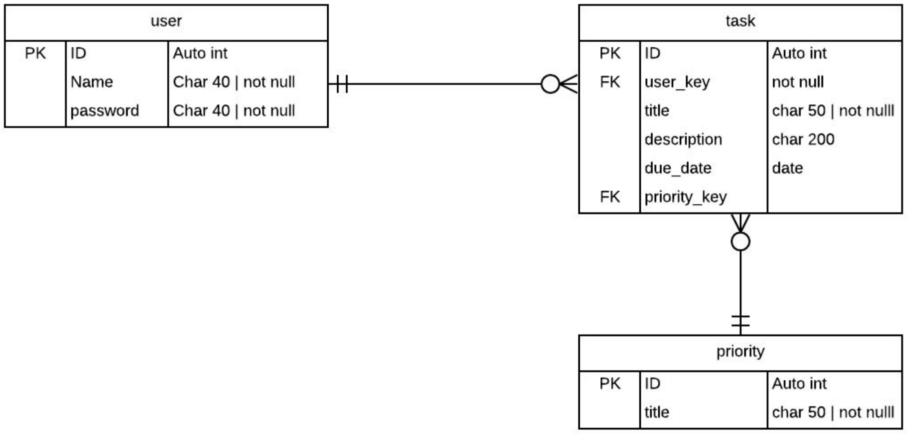
Methodology

- Explain the algorithm you designed for the project.
    - We didn't use too many algorithms outside some very basic searches. Most of the complexity we faced is dealing with instantiation of classes from large libraries.
- Describe the implementation process. What did you implement?
    - Because our app consists of a basic GUI and a database, we decided to implement the MVC (model, view, controller) structure. The idea behind this structure is that you split up your code base into 3 major parts. One controlling how the program looks at run time. This view section is created by the classes that are within the tornado FX library. One controlling the logic, the controller, behind the program, this is handled by our Kotlin code base we created from scratch. It mostly consists of opening views at the right time and moving/Manipulating data around. And the model dealing with getting any data needed from the database. This part is handled by a combination of the Kotlin classes we created, maven implementation, and PostgreSQL. This section of our implementation is also where AWS (Amazon Web Services) comes into play. So now instead of our database being stored locally, it's stored on a server over the internet. This in theory would allow multiple different users at the same time to interact with the service.

Technology Stack



Database v_1
Todo app

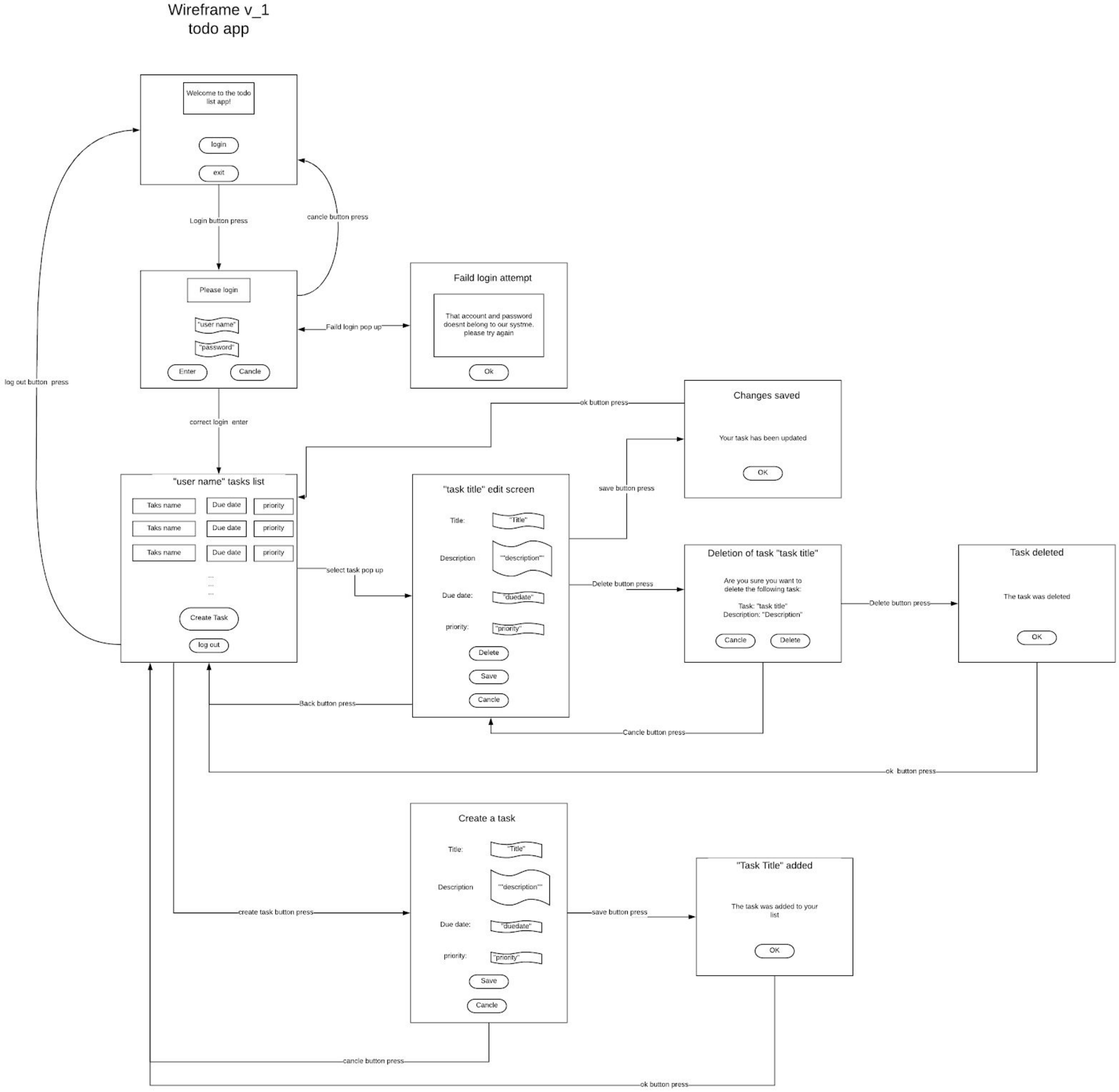


user		
PK	ID	Auto int
	Name	Char 40 | not null
	password	Char 40 | not null

task		
PK	ID	Auto int
FK	user_key	not null
	title	char 50 | not nulll
	description	char 200
	due_date	date
FK	priority_key	

priority		
PK	ID	Auto int
	title	char 50 | not nulll

- WireFrame



Demo and Results

- Describe the tests you ran to determine if your program accurately reaches the stated goal.
  - We based all our tests off our user stories, working towards implementing the user stories in the program. Once a user story is done, we will then

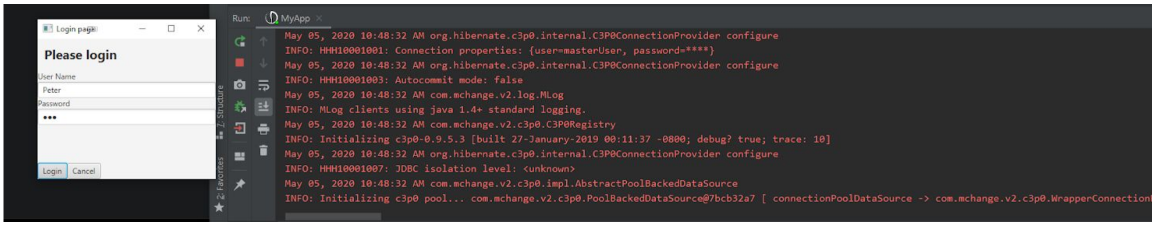
test that user story. Our overall testing starts by testing ‘Features’ of that user story. We test an expected feature of a user story vs its actual outcome. Once all the features are tested, we then have a bug report and can fix the bus within that report

- Testing table

#	Story	action	outcome	result	commits	status
1	1	The user enters a new task	The user can enter a new task in a form	The user can enter in new tasks into the system	works	pass
2	1	A New task is saved in the system	A new task is saved in the database			Untested
3	1	A user can edit their old tasks	Can change any field in an old task	The user can open task into an edit menu	works	pass
4	1	Edits to a task are saved in the system	Any changes to a task is saved in the database			Untested
5	2	A User can view a list of all tasks assigned to that user	Can see all tasks that are assigned to that user	The user sees tasks assigned to them	works	pass
6	2	A user can get a quick glance of each task	Each task is displayed in a way to show information about that task	The list view of all task are show and updates properly	Works	pass
7	2	User does not see old tasks	A user does not see tasks that have been deleted from the system.			Untested
8	3	A user can delete unwanted tasks	A user can delete any task assigned to them			Untested

9	3	A user is warned before deleting	A warning message of some type is shown before deleting	There no warning currently	Add some warning alerts to let them know they are about to delete an item	fail
10	4	A user is prompted to log into the system	There is a log in screen when the user launches the app	The login screen is show on first boot	works	pass
11	4	An authorized user can enter the system	A user has to have the right username and pass word to see tasks assigned with that user			Untested
12	4	An unauthorized user can not enter the system	A user with the wrong name and password is not allowed to enter the system			Untested
13	5	Different users are kept track by the system	There can be multiple users on the system and each have their own set of tasks			Untested
14	5	New users can be created	New users can be created with in the app	No way in the app to create a new users	We need to create a ‘new user’ portal	fail
15	5	Different users can use the app at the same time	Different users can access the database at the same time to view tasks on the system			Untested
16	6	A user can sort their tasks by title	A user can view all tasks assorted by alphabetical ordering	There is no way to sort by	Need to add a way to sort by the title of the task	fail
17	6	A user can sort their tasks by date due	A user can sort their task by			Untested

			when the due date is the closets to current date			
18	6	A user can sort their task by priority	A user can sort their task by which task is the most import			Untested
19	7	A user can set a due date for tasks	A user can set when any task is set to be due	Task have the option to add a due date to them	works	pass
20	7	A user can set a priority level for a task	A user can set a priority tag for each tag showing how important it is	Tasks have an option to add a priority level to them	works	pass

- Program Run Time Screenshots
    - Logging in and system accessing database on AWS
        - 

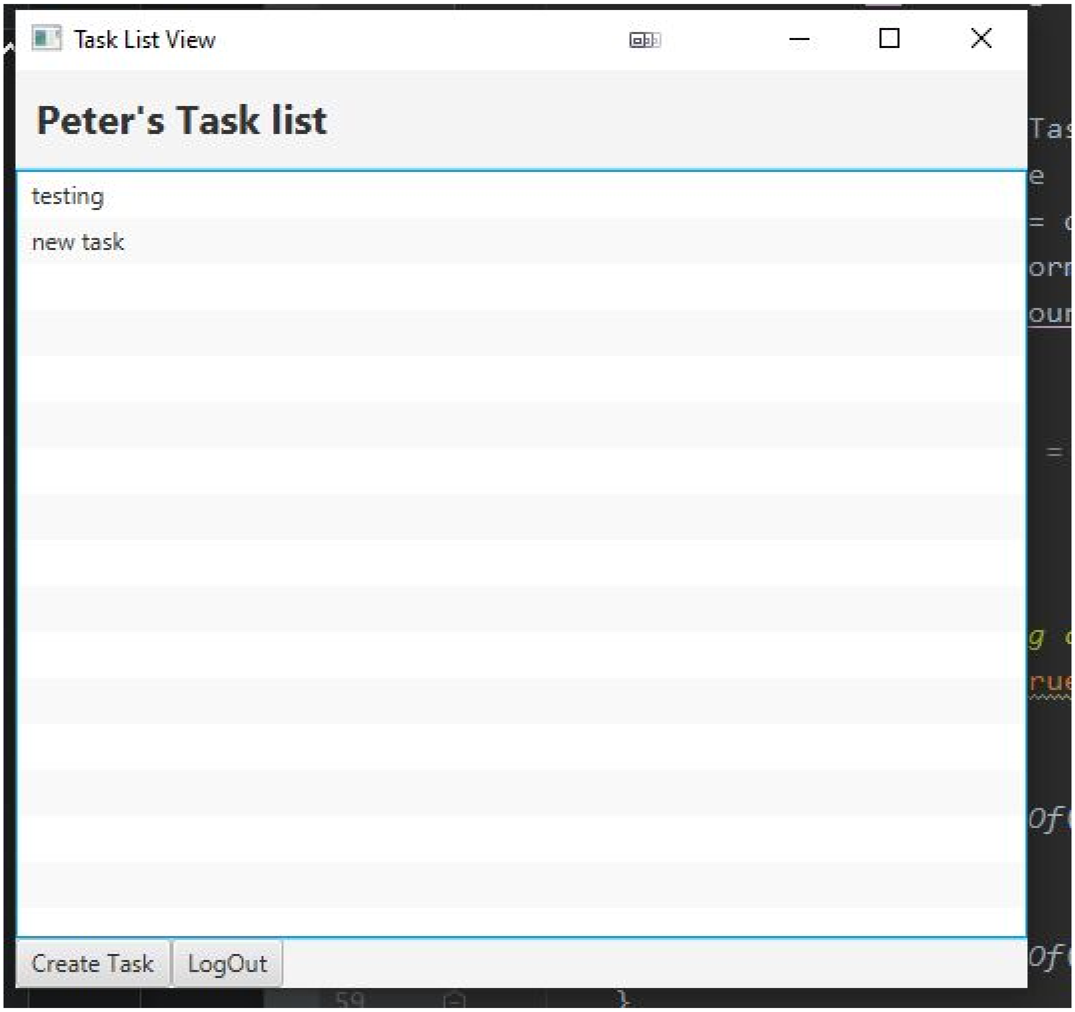
Please login

User Name: Peter

Password: •••

```
May 05, 2020 10:48:32 AM org.hibernate.c3p0.internal.C3P0ConnectionProvider configure
INFO: HHH10001001: Connection properties: {user=masterUser, password=****}
May 05, 2020 10:48:32 AM org.hibernate.c3p0.internal.C3P0ConnectionProvider configure
INFO: HHH10001003: Autocommit mode: false
May 05, 2020 10:48:32 AM com.mchange.v2.log.MLog
INFO: MLog clients using java 1.4+ standard logging.
May 05, 2020 10:48:32 AM com.mchange.v2.c3p0.C3P0Registry
INFO: Initializing c3p0-0.9.5.3 [built 27-January-2019 00:11:37 -0800; debug? true; trace: 10]
May 05, 2020 10:48:32 AM org.hibernate.c3p0.internal.C3P0ConnectionProvider configure
INFO: HHH10001007: JDBC isolation level: <unknown>
May 05, 2020 10:48:32 AM com.mchange.v2.c3p0.impl.AbstractPoolBackedDataSource
INFO: Initializing c3p0 pool... com.mchange.v2.c3p0.PoolBackedDataSource@7bcb32a7 [ connectionPoolDataSource -> com.mchange.v2.c3p0.WrapperConnection
```

    - User seeing their list of tasks

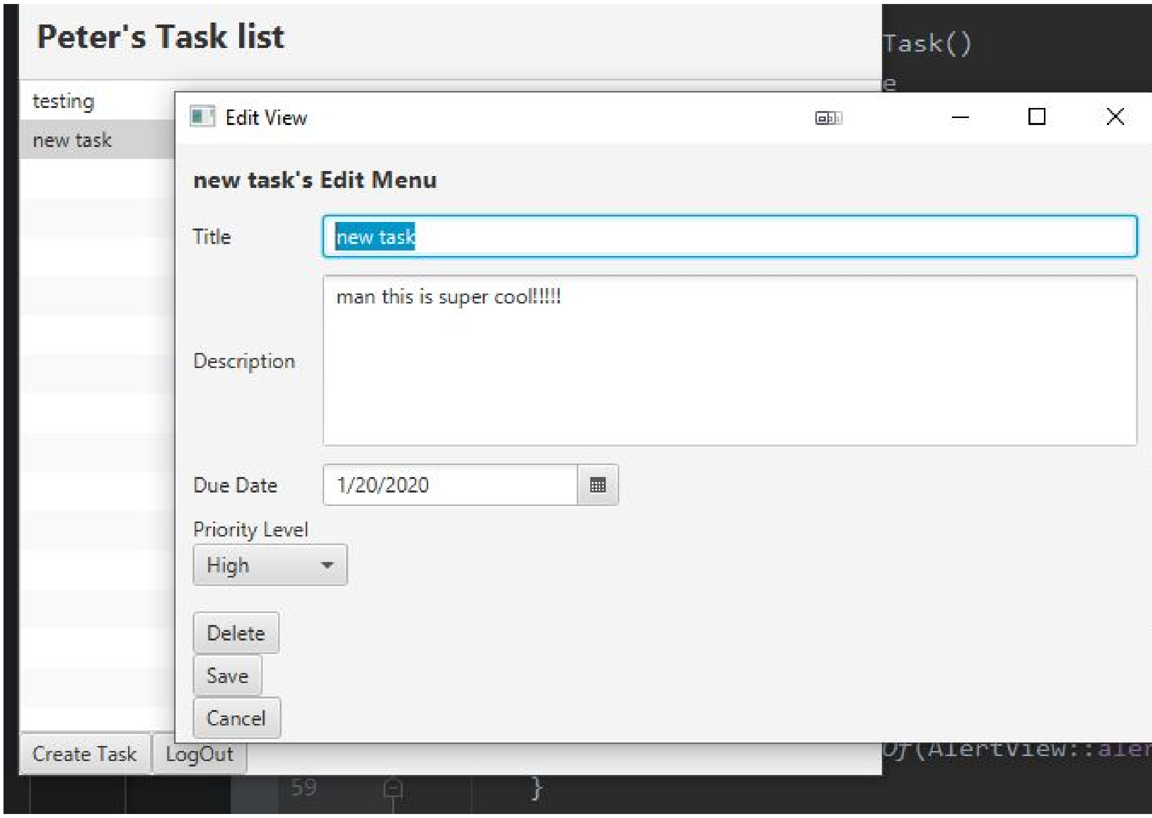
        - 

          Peter's Task list

          testing

          new task

    - Editing a task



Peter's Task list

testing

new task

Edit View

new task's Edit Menu

Title: new task

Description: man this is super cool!!!!!

Due Date: 1/20/2020

Priority Level: High

Discussion

- What did you learn by doing this project?
  - That you can tackle new technologies/ languages much easier than we had thought. Going into this project, we didn't know much about Kotlin, AWS, or tornado FX. In a few weeks time, we are able to learn and use these technologies. We are not masters of these technologies currently, but we are able to understand them and use them to create simple programs.
- What was the most difficult part in completing this assignment?
  - Finding examples on how to use the tornado FX library. Because the library is relatively new, finding examples of how to use some classes within the library was posing challenges. If we ran into problems on how to handle a view, we had to look at java FX, which tornado FX is built on top

of, and take a guess and how some views work in Kotlin. Or implement our problem in another way.

- What did you like about the assignment? What you did not like?
    - We like the idea of picking whatever language we wanted to make a program in. picking a new language we wanted to learn was pretty fun and a great learning experience.
    - We didn't like the time frame for the assignment. It would have been a better experience if the project was given to us earlier in the semester. This is because learning a new language and creating something in it did take more time than we thought. The project is at the point now where we either have to cram or take out features of the program. It is possible that the coronavirus outbreak could have screwed our opinions on this matter. Considering how out of whack everyone's schedule got during this time.
- What would you have done differently if you were starting the project again?
    - Possibly scaled back the amount of new technologies the team had to learn. In our case, we all had to figure out 4-5 new technologies from AWS to tornado FX. This added to our time and the complexity of the application. But considering how great of a learning experience it was and our app increased in its ‘real world validity’, I still think it's the best we went this route.