Awesome app

Cosmin Bucur

Oct 2021

Agenda

- objectives
- deliverables (trainer, student)
- startup methodology
- HR
  - hiring process, team, roles
  - achievement system
- collaboration
  - ask questions live
  - ask questions remote
- plan
  - agile ceremonies
  - agile artifacts
- design
- develop
  - infrastructure
  - backend
  - frontend
- demo final project

A Team Wild production

D oy

Cosmin Bucur

Product Owner

Agile Coach

Member 1	Member 2	Member 3	Member 3
Senior dev	Regular 2 dev	Regular 1 dev	Junior dev
Tech lead			

Objectives

for final project

- Work in a dev environment
- Learn how to learn
- Learn to structure knowledge
- Build the final project
- Learn how to Demo

Deliverables

in project

- plan
- design
- devops
- infrastructure
- code
- learn
- sample

Trainer deliverables

Methodology

- achievements doc - a grading system based on the gamification concept
- STEPS.md - keep track of global priorities
- issue system.md - list of issue categories
- ISSUES.md - keep track of issues
- cosmin bucur - awesome-app - a high quality presentation of the final project

Trainer deliverables

Plan

- agile.drawio - diagram that defines the agile concepts
- BACKLOG.md - trainer backlog
- BOARD.md - trainer board
- sprints sample.md - sprint goals and dates
- tasks backend.md - set of backend technical tasks
- tasks frontend.md - set of frontend technical tasks
- definition of ready.md - explains when a story is ready for development
- definition of done.md - explains when a story is fully implemented

Trainer deliverables

Plan

- user story sample.md - how to write a user story
- task sample.md - how to write a task
- bug sample.md - how to write a user story
- backlog sample.md - how to create a backlog
- board sample.md - how to create a sprint board

Trainer deliverables

Design

- design.drawio - trainer design
- routing.drawio - trainer routing plan

Trainer deliverables

Devops

- CI-CD.drawio - diagram with a continuous integration / development pipeline
    - software, commands and artifacts in a CI-CD pipeline
- monitor.drawio - how to monitor application (metrics and logs)

Trainer deliverables

Infrastructure

- technology stack sample.drawio - diagram with the selected technologies
- architecture sample.drawio - diagram with a real application architecture
- database model sample.drawio - the database model diagram
- database model sample.mwb - the database model diagram in mysql workbench

Learn how to learn

Trainer project

- concepts - what to learn and in which order
    - list of markdown files with the essential concepts in the natural order
- diagrams - diagrams using the Building Blocks™ approach
    - remember concepts and create connections
- samples - increase development speed
    - application properties, logger config
- code examples - how to translate concepts in code
    - code examples for all training modules

Student deliverables

- README.md - project info (project name, github repo, team info)
- design.drawio - the design of the application
- routing.drawio - the routing plang
- backlog.md - the list of issues
- user stories - the detail user stories in markdown format
  - (TODO, PROGRESS, DONE)
- board.md - the active sprint issues and state
- database model.mwb - the database model diagram in mysql workbench
- code - the final project in a github repository
- demo - the project presentation slides

Startup

methodology

- HR
- Collaboration
- Plan
- Design
- Develop
- Test
- Release

HR

HR

steps

- hire people by self assessment
- define roles
- split teams

animal-app	mica-piata	my-home	welness-app	cosmin
Soul	Ares	Enigma	Vega	product owner
madalina	vlad	ana	florentina	scrum master
mihaela	ionut	george	alin	agile coach
dana		giani	ioan	

cake-shop	hr-app	patent-app	task-app	travel-app
brebenel	david	stefan	cristi	catalina

intern	3000
junior	4000
regular	6000
regular 2	10000
senior	15000

designer	5000
tester	5000

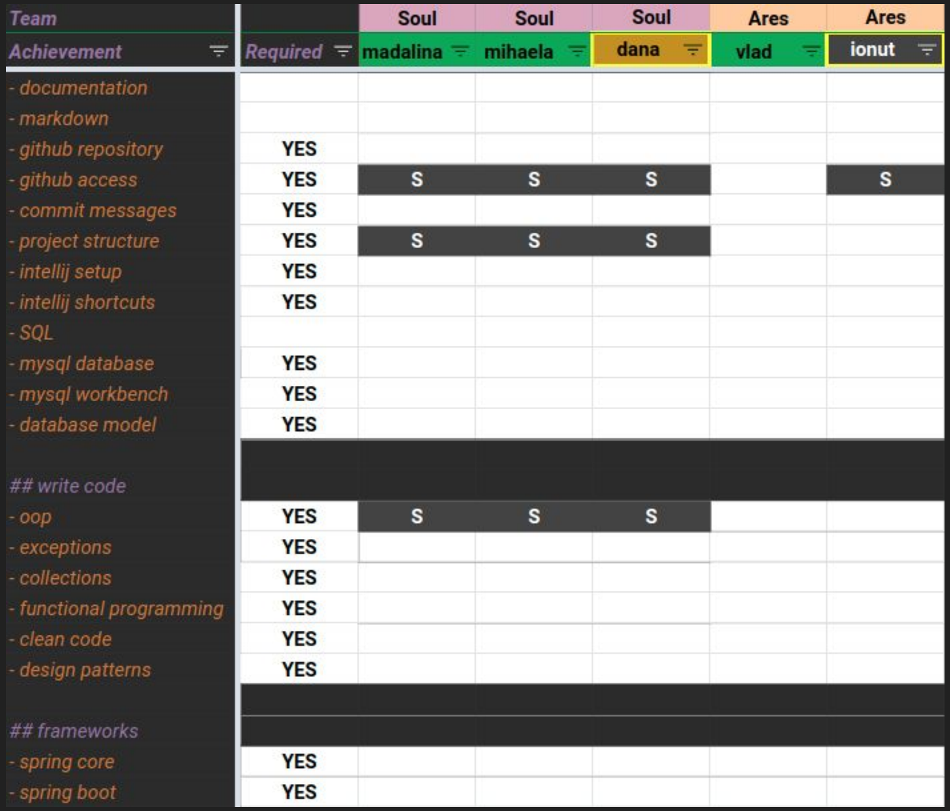
product owner	8000
scrum master	8000
agile coach	15000

Achievement system

- essential knowledge
- backend achievements
- frontend achievements
- soft skills
- technical skills

Grading

- S - senior
- R2 - regular 2
- R1 - regular 1
- J - junior



Team		Soul	Soul	Soul	Ares	Ares
Achievement	Required	madalina	mihaela	dana	vlad	ionut
- documentation						
- markdown						
- github repository	YES					
- github access	YES	S	S	S		S
- commit messages	YES					
- project structure	YES	S	S	S		
- intellij setup	YES					
- intellij shortcuts	YES					
- SQL						
- mysql database	YES					
- mysql workbench	YES					
- database model	YES					
## write code						
- oop	YES	S	S	S		
- exceptions	YES					
- collections	YES					
- functional programming	YES					
- clean code	YES					
- design patterns	YES					
## frameworks						
- spring core	YES					
- spring boot	YES					

Asking questions

The issues fall in these categories

1 plan

- backlog
- design

2 environment

- intellij
- git
- maven
- mysql workbench

3 backend

- java
- hibernate
- spring

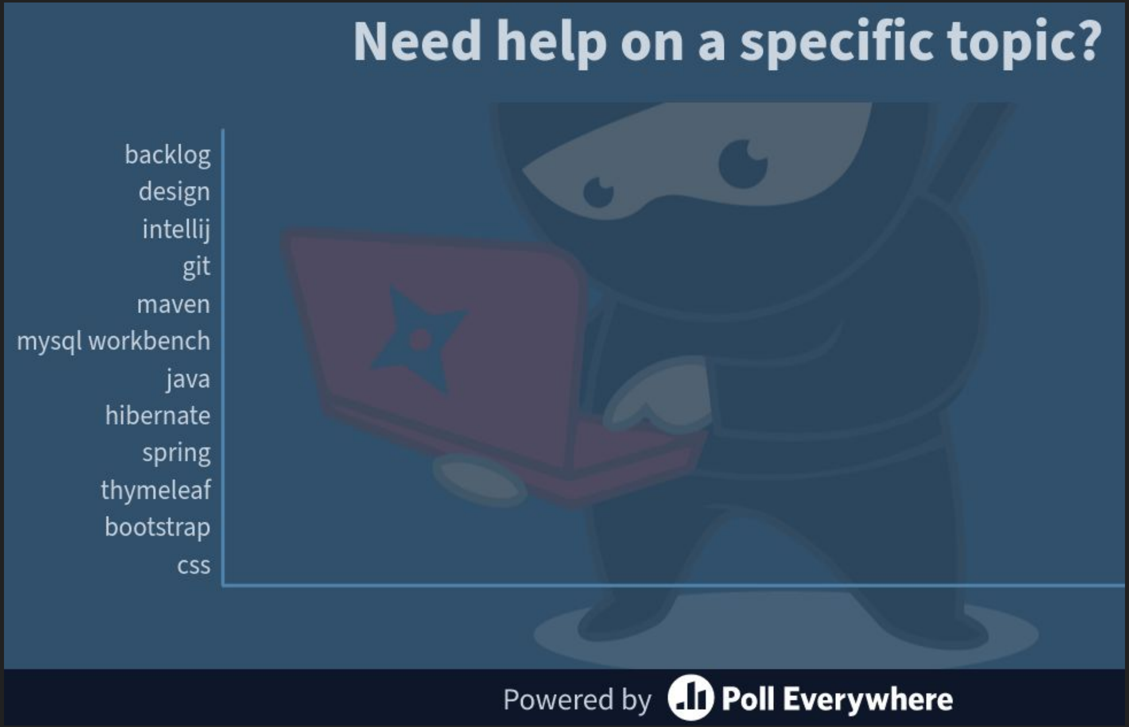
4 frontend

- thymeleaf
- boostrap
- css

Live questions

The student uses poll everywhere to select his issue category

https://PollEv.com/cosminbucur951

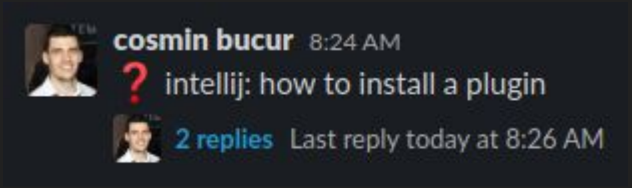

Need help on a specific topic?

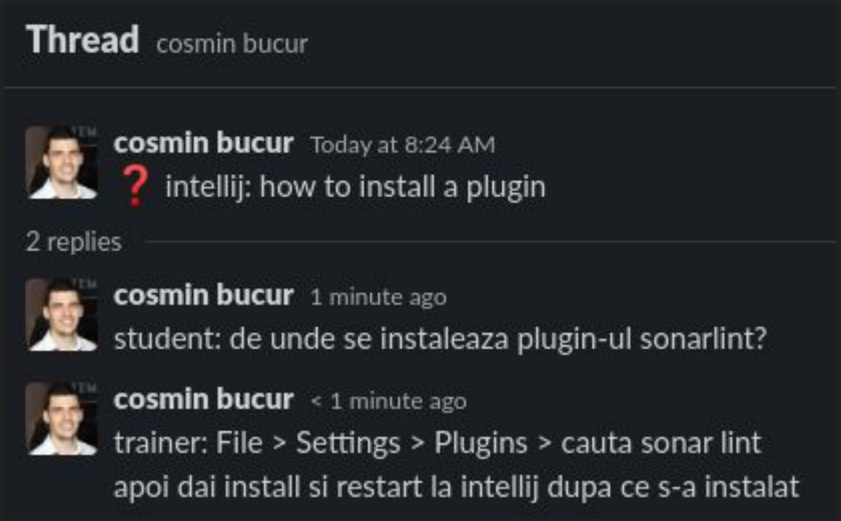
- backlog
- design
- intellij
- git
- maven
- mysql workbench
- java
- hibernate
- spring
- thymeleaf
- bootstrap
- css

Remote questions

Create Slack thread channel #exercitii

Use threads in slack to group messages regarding a specific issue


cosmin bucur 8:24 AM
❓ intellij: how to install a plugin
2 replies Last reply today at 8:26 AM


Thread cosmin bucur

cosmin bucur Today at 8:24 AM
❓ intellij: how to install a plugin

2 replies

cosmin bucur 1 minute ago
student: de unde se instaleaza plugin-ul sonarlint?

cosmin bucur < 1 minute ago
trainer: File > Settings > Plugins > cauta sonar lint
apoi dai install si restart la intellij dupa ce s-a instalat

Issue system

The trainer uses the ISSUES.md file to track all issues

Collaboration

Collaboration

Slack for chat

Zoom for Break Rooms

Poll Everywhere to keep track of QA

Plan

Agile

ceremonies

- Refinement
- Planning
- Sprint
- Demo

Refinement

steps

- define purpose
- define domain (entities)
- define epics
- define user stories
- refine user stories
- estimate effort

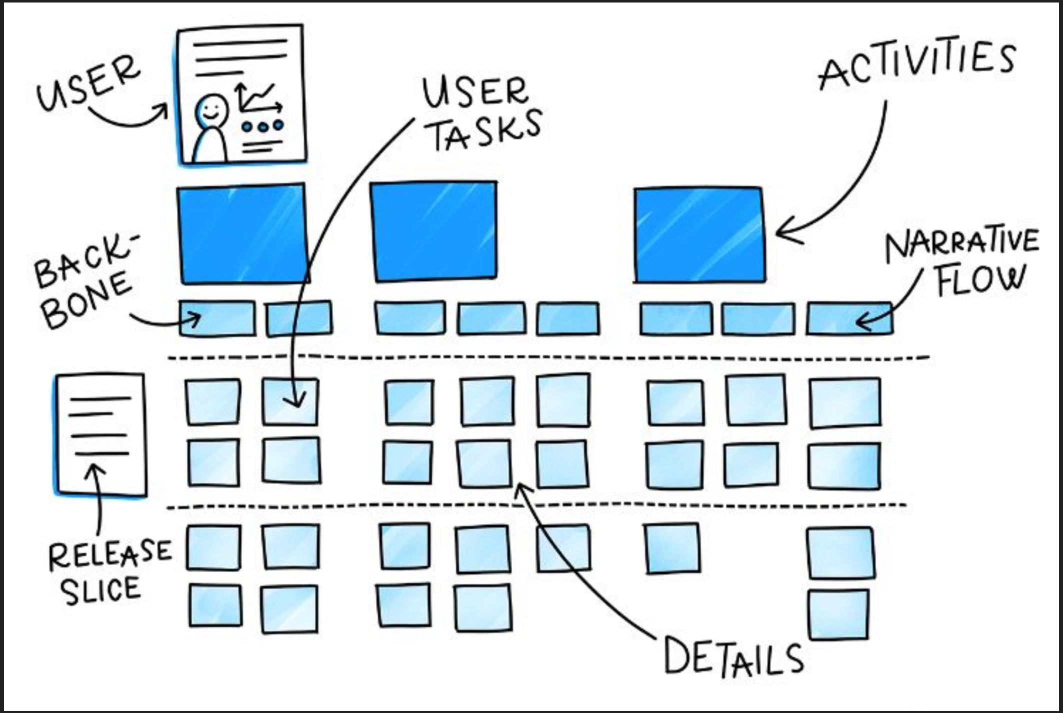
Refinement

artifacts

- Backlog
  - Epic
  - User Story
    - Acceptance Criteria
    - Estimation
  - Task
  - Bug
- Definition of Ready
- Definition of Done
- Design
- Routing plan

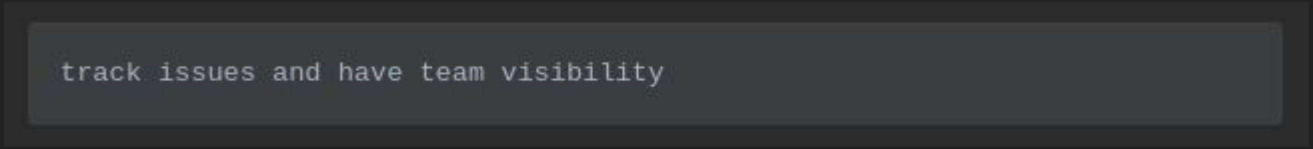
Story mapping

Used story mapping to describe the main functionality of the app



Define purpose

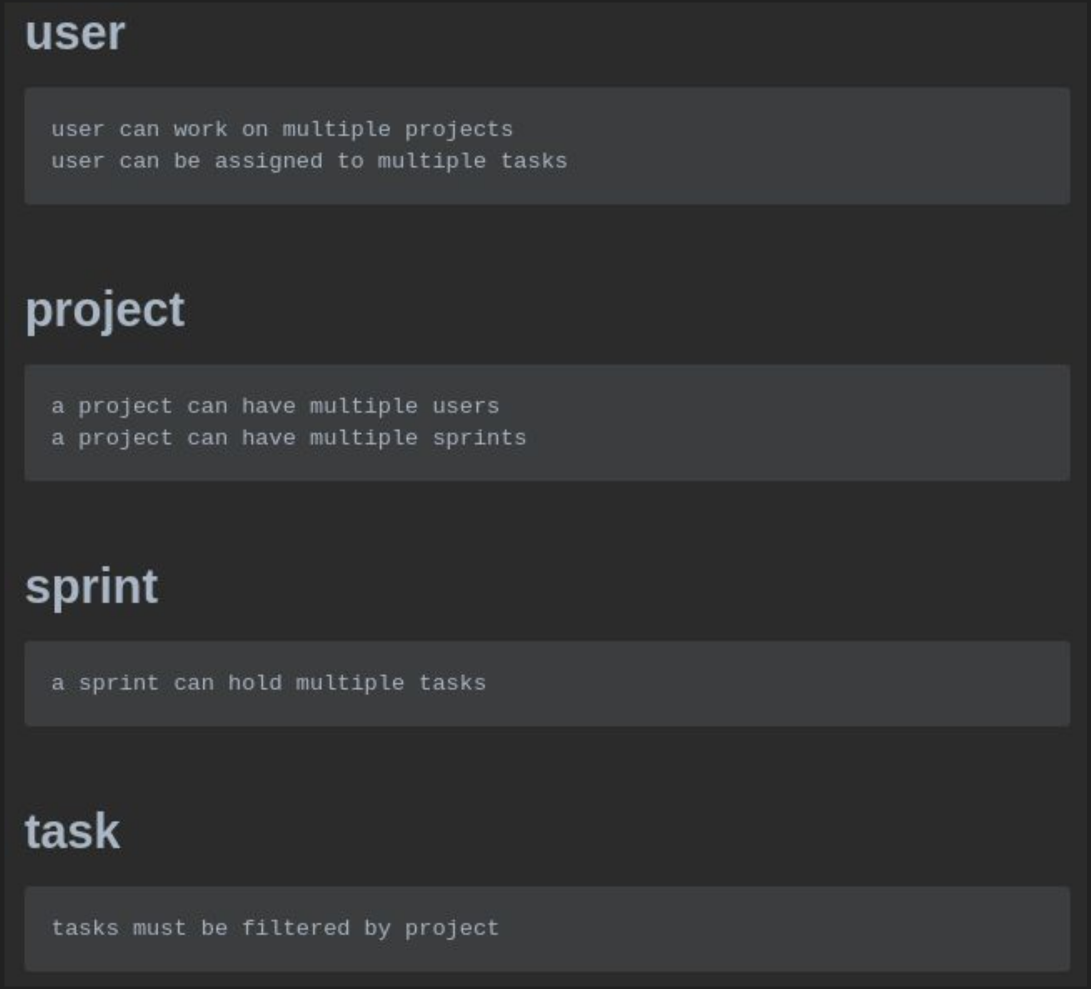
Task management can’t be easier than this!



```
track issues and have team visibility
```

Define logo

Define entities



user

```
user can work on multiple projects
user can be assigned to multiple tasks
```

project

```
a project can have multiple users
a project can have multiple sprints
```

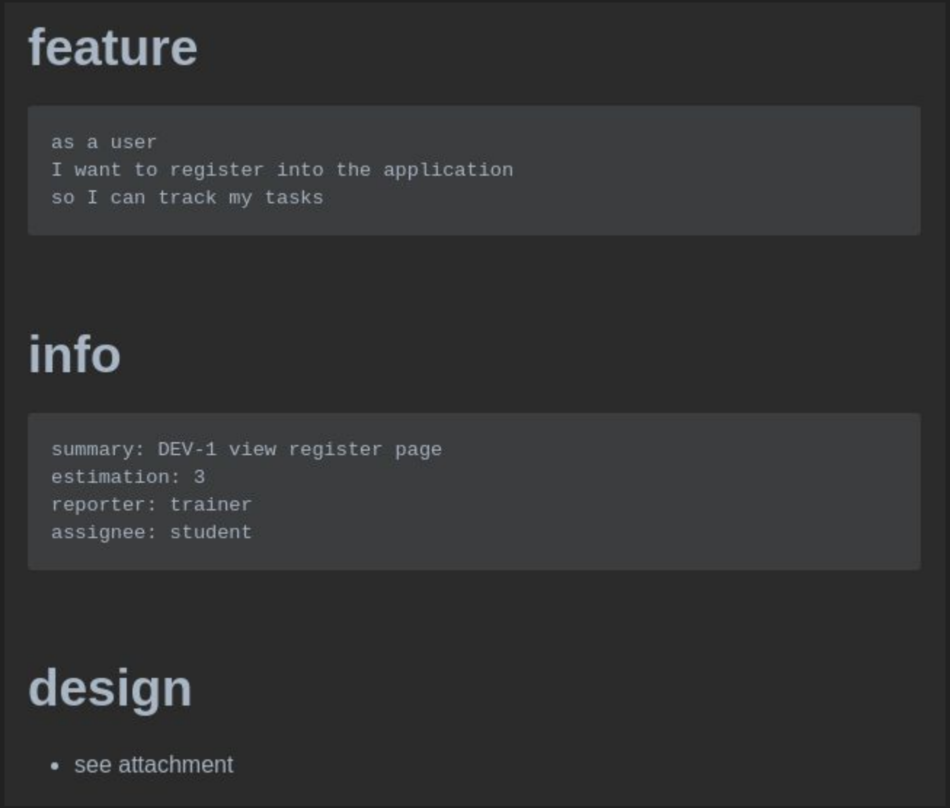
sprint

```
a sprint can hold multiple tasks
```

task

```
tasks must be filtered by project
```

User story sample



feature

```
as a user
I want to register into the application
so I can track my tasks
```

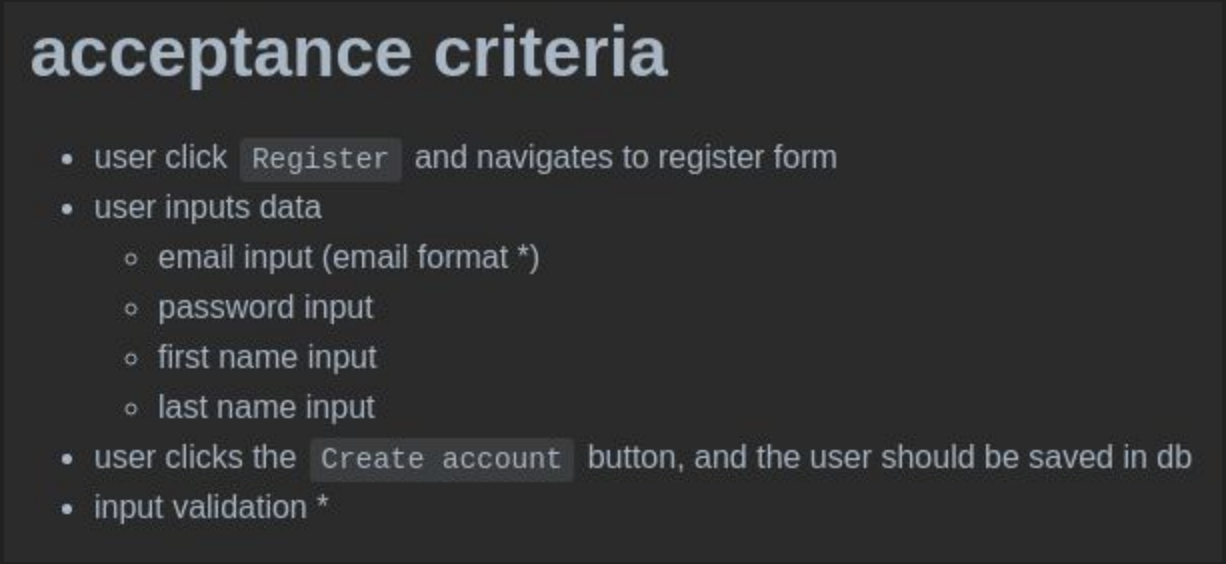
info

```
summary: DEV-1 view register page
estimation: 3
reporter: trainer
assignee: student
```

design

- see attachment

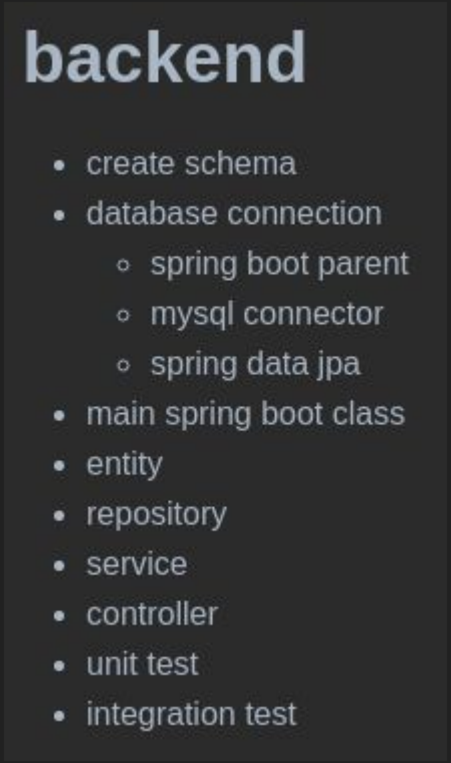
User story sample



acceptance criteria

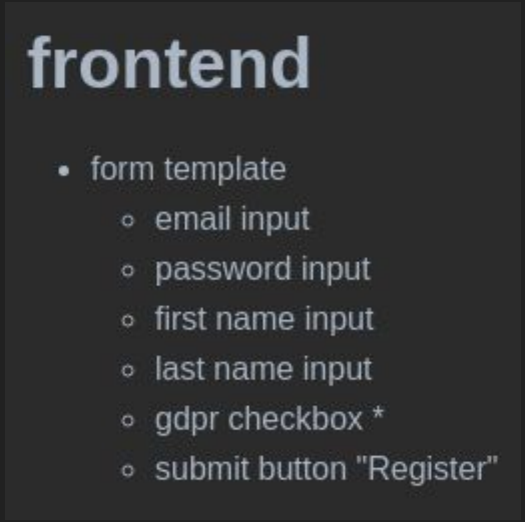
- user click `Register` and navigates to register form
- user inputs data
    - email input (email format *)
    - password input
    - first name input
    - last name input
- user clicks the `Create account` button, and the user should be saved in db
- input validation *

User story sample



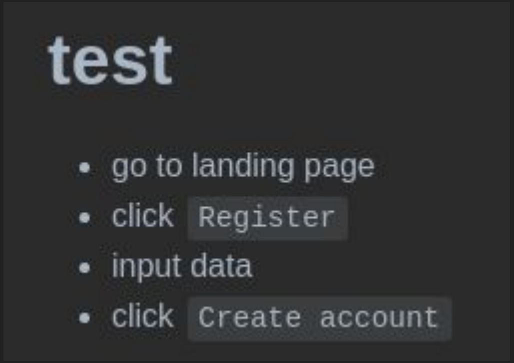
backend

- create schema
- database connection
  - spring boot parent
  - mysql connector
  - spring data jpa
- main spring boot class
- entity
- repository
- service
- controller
- unit test
- integration test



frontend

- form template
  - email input
  - password input
  - first name input
  - last name input
  - gdpr checkbox *
  - submit button "Register"



test

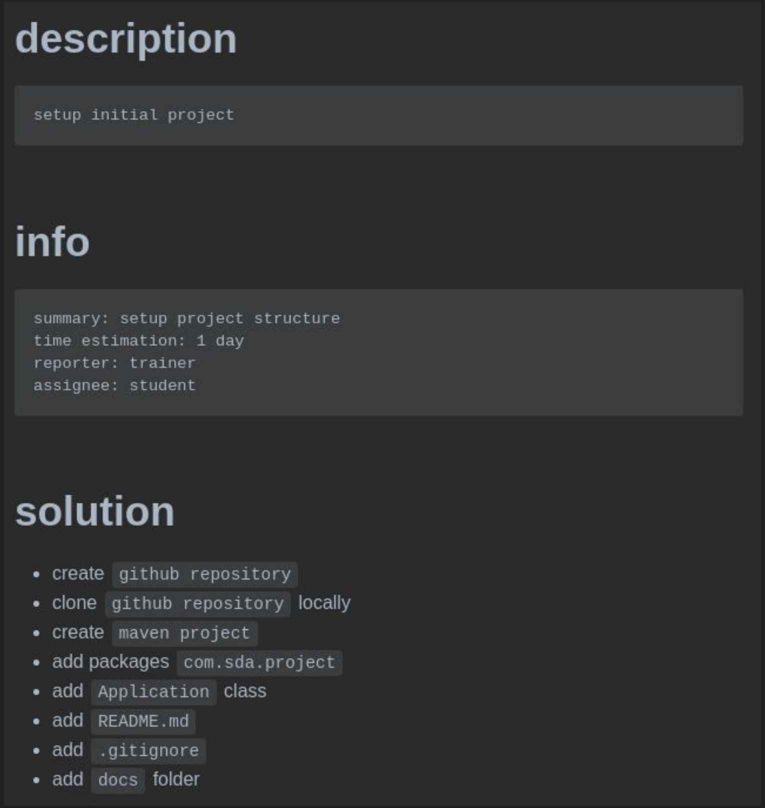
- go to landing page
- click `Register`
- input data
- click `Create account`

Definition of

Ready

- complete design
- acceptance criteria
- no external blockers

Task sample



description

```
setup initial project
```

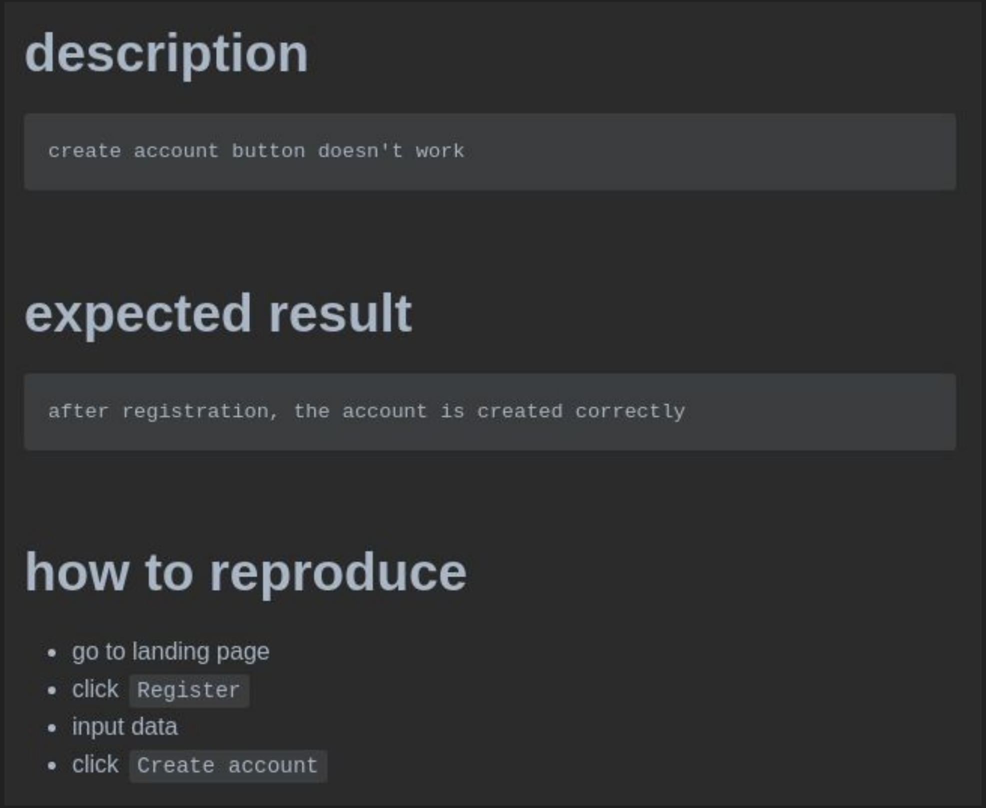
info

```
summary: setup project structure
time estimation: 1 day
reporter: trainer
assignee: student
```

solution

- create `github repository`
- clone `github repository` locally
- create `maven project`
- add packages `com.sda.project`
- add `Application` class
- add `README.md`
- add `.gitignore`
- add `docs` folder

Bug sample



description

```
create account button doesn't work
```

expected result

```
after registration, the account is created correctly
```

how to reproduce

- go to landing page
- click `Register`
- input data
- click `Create account`

Definition of

Done

- run unit tests
- run integration tests
- run app locally
- create pull request
- merge request

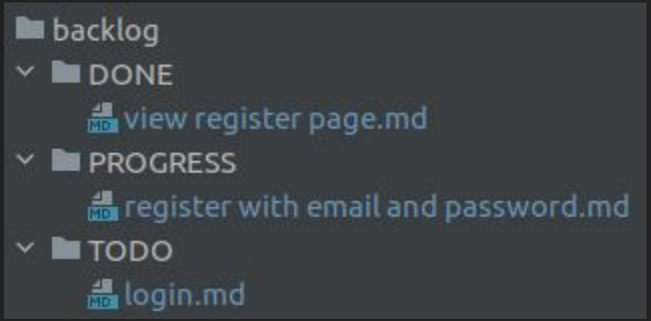
Epics

definition

- Onboarding
- Manage projects
- Manage sprints
- Manage tasks
- Manage users

Task management

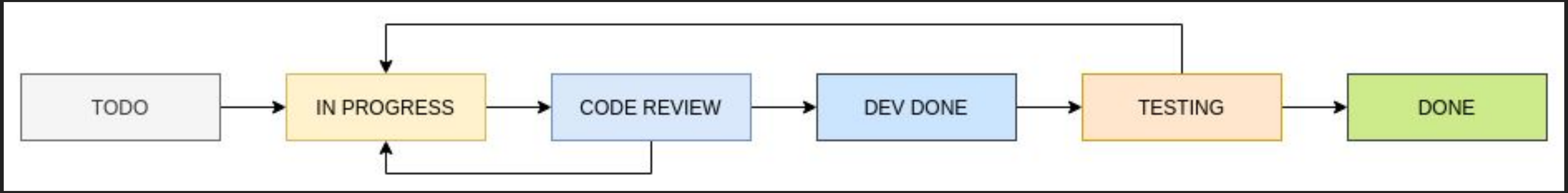
Used markdown to track tasks



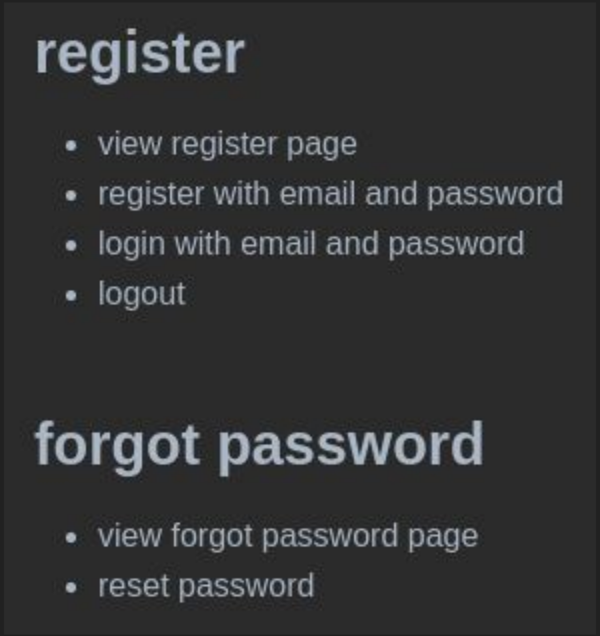
```
backlog
⌄ DONE
    view register page.md
⌄ PROGRESS
    register with email and password.md
⌄ TODO
    login.md
```

Agile Workflow

Used this task management workflow



Backlog



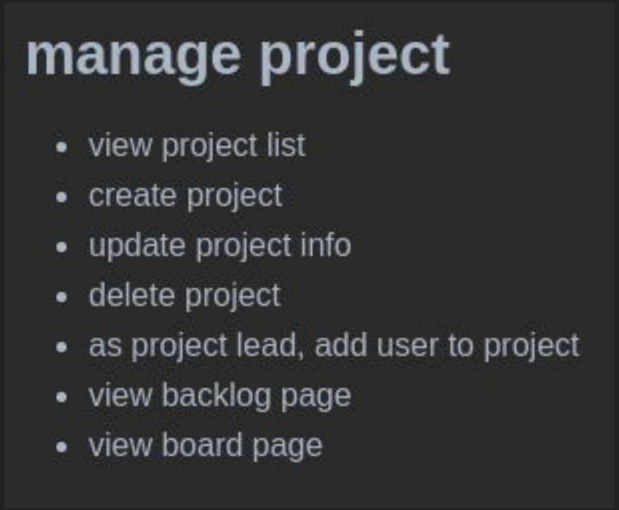
register

- view register page
- register with email and password
- login with email and password
- logout

forgot password

- view forgot password page
- reset password

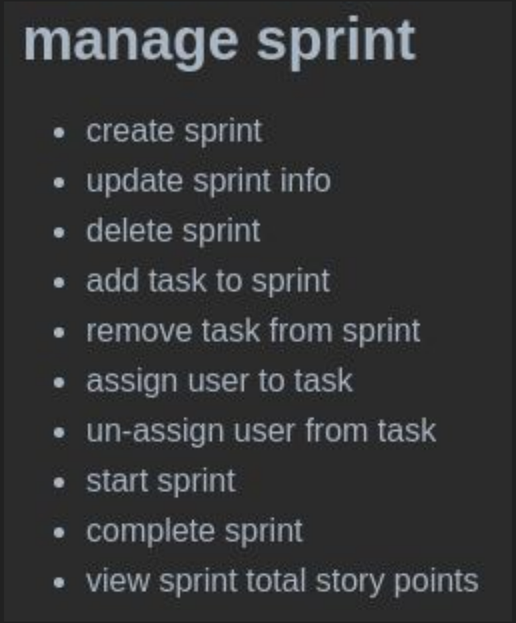
Backlog



manage project

- view project list
- create project
- update project info
- delete project
- as project lead, add user to project
- view backlog page
- view board page

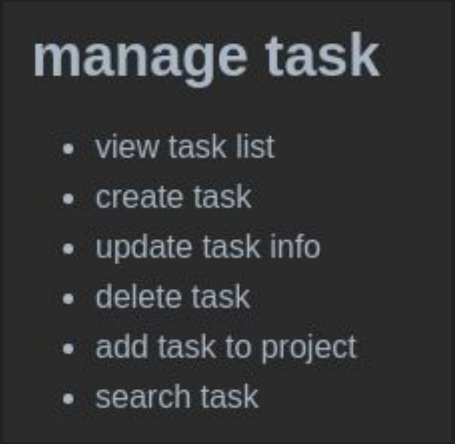
Backlog



manage sprint

- create sprint
- update sprint info
- delete sprint
- add task to sprint
- remove task from sprint
- assign user to task
- un-assign user from task
- start sprint
- complete sprint
- view sprint total story points

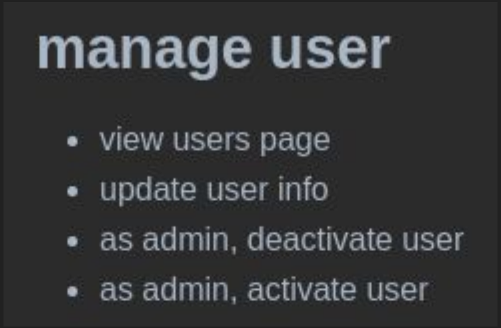
Backlog



manage task

- view task list
- create task
- update task info
- delete task
- add task to project
- search task

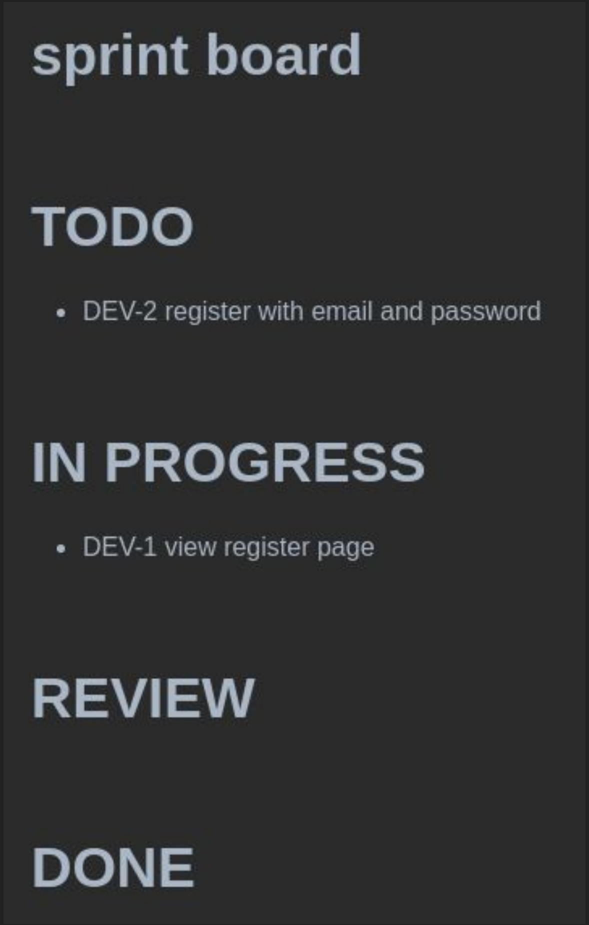
Backlog



manage user

- view users page
- update user info
- as admin, deactivate user
- as admin, activate user

Board



sprint board

TODO

- DEV-2 register with email and password

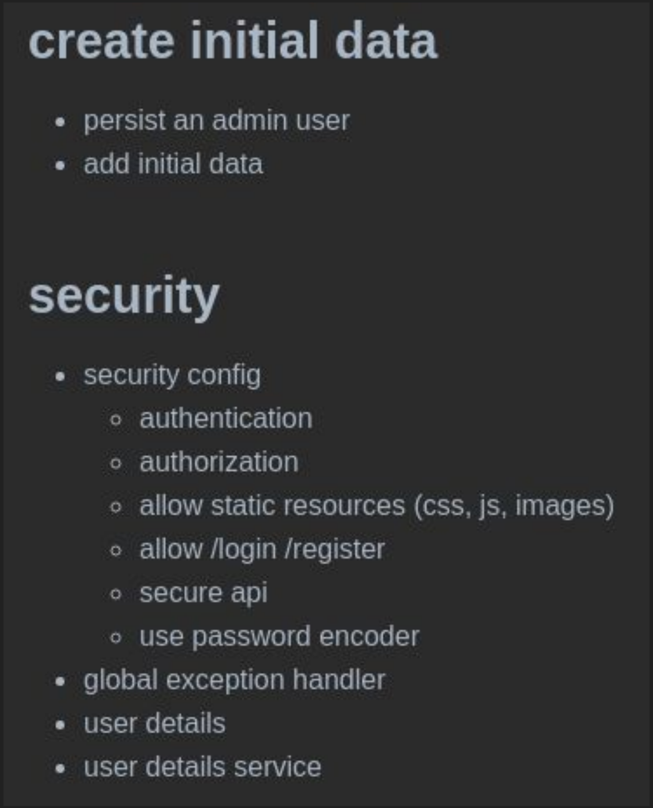
IN PROGRESS

- DEV-1 view register page

REVIEW

DONE

Backend tasks



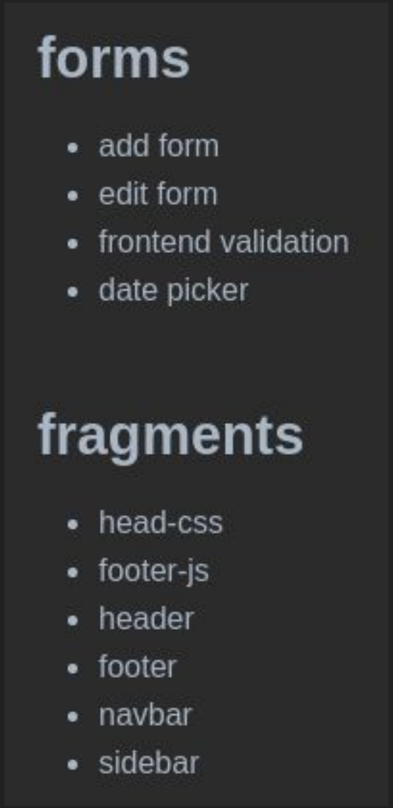
create initial data

- persist an admin user
- add initial data

security

- security config
  - authentication
  - authorization
  - allow static resources (css, js, images)
  - allow /login /register
  - secure api
  - use password encoder
- global exception handler
- user details
- user details service

Frontend tasks



forms

- add form
- edit form
- frontend validation
- date picker

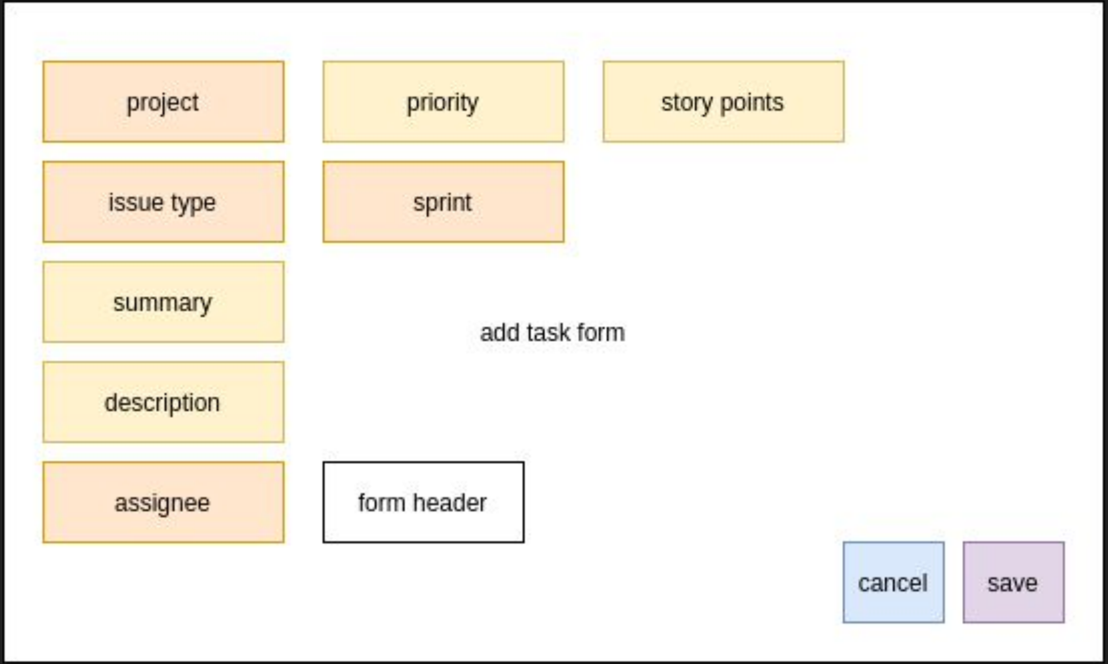
fragments

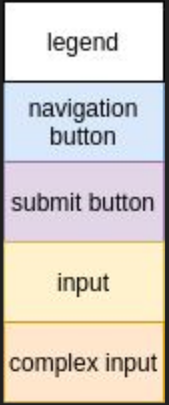
- head-css
- footer-js
- header
- footer
- navbar
- sidebar

Design

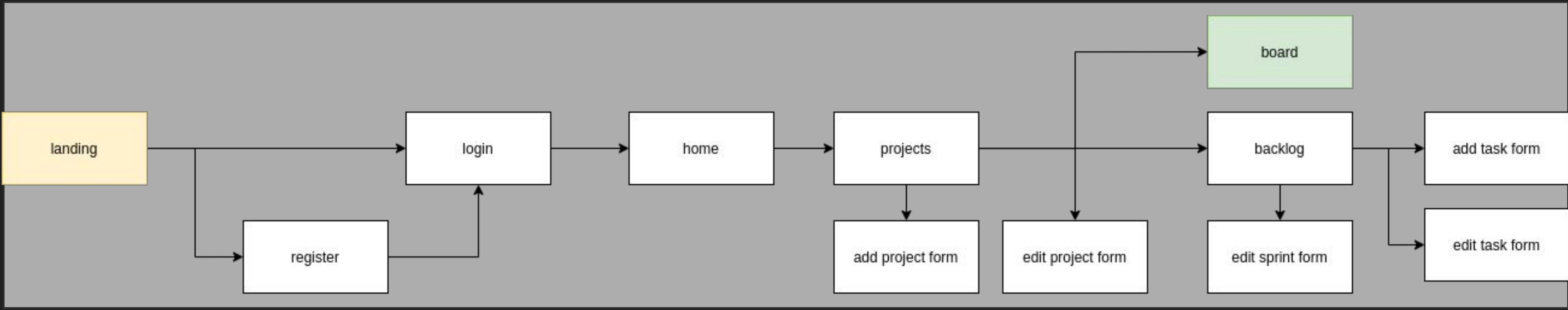
Design

Used https://app.diagrams.net/ to sketch the design





Routing plan



Planning

steps

- define sprints
- set sprint goals
- add stories to sprints

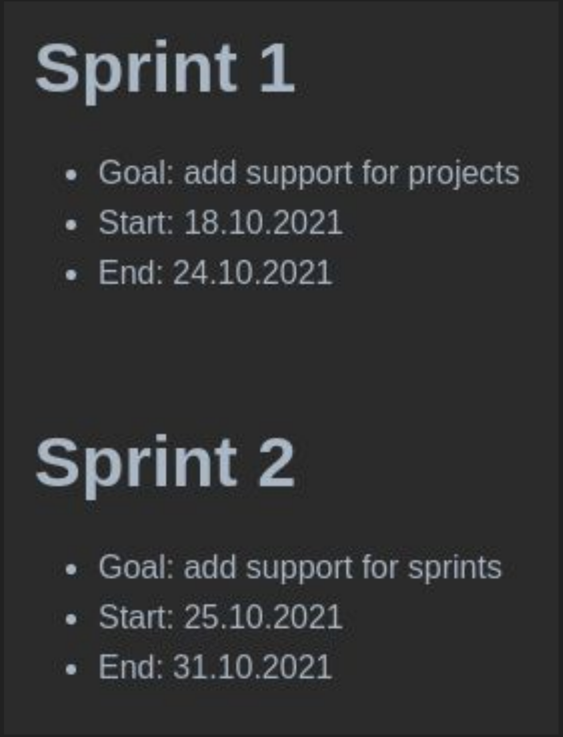
Sprints

Prepared

5 days x 3 hours

Developed

4 sprints x 1 week

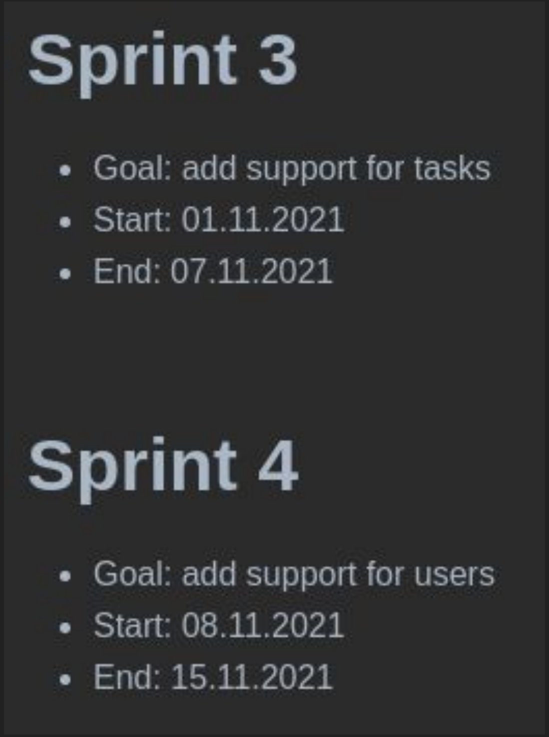


Sprint 1

- Goal: add support for projects
- Start: 18.10.2021
- End: 24.10.2021

Sprint 2

- Goal: add support for sprints
- Start: 25.10.2021
- End: 31.10.2021



Sprint 3

- Goal: add support for tasks
- Start: 01.11.2021
- End: 07.11.2021

Sprint 4

- Goal: add support for users
- Start: 08.11.2021
- End: 15.11.2021

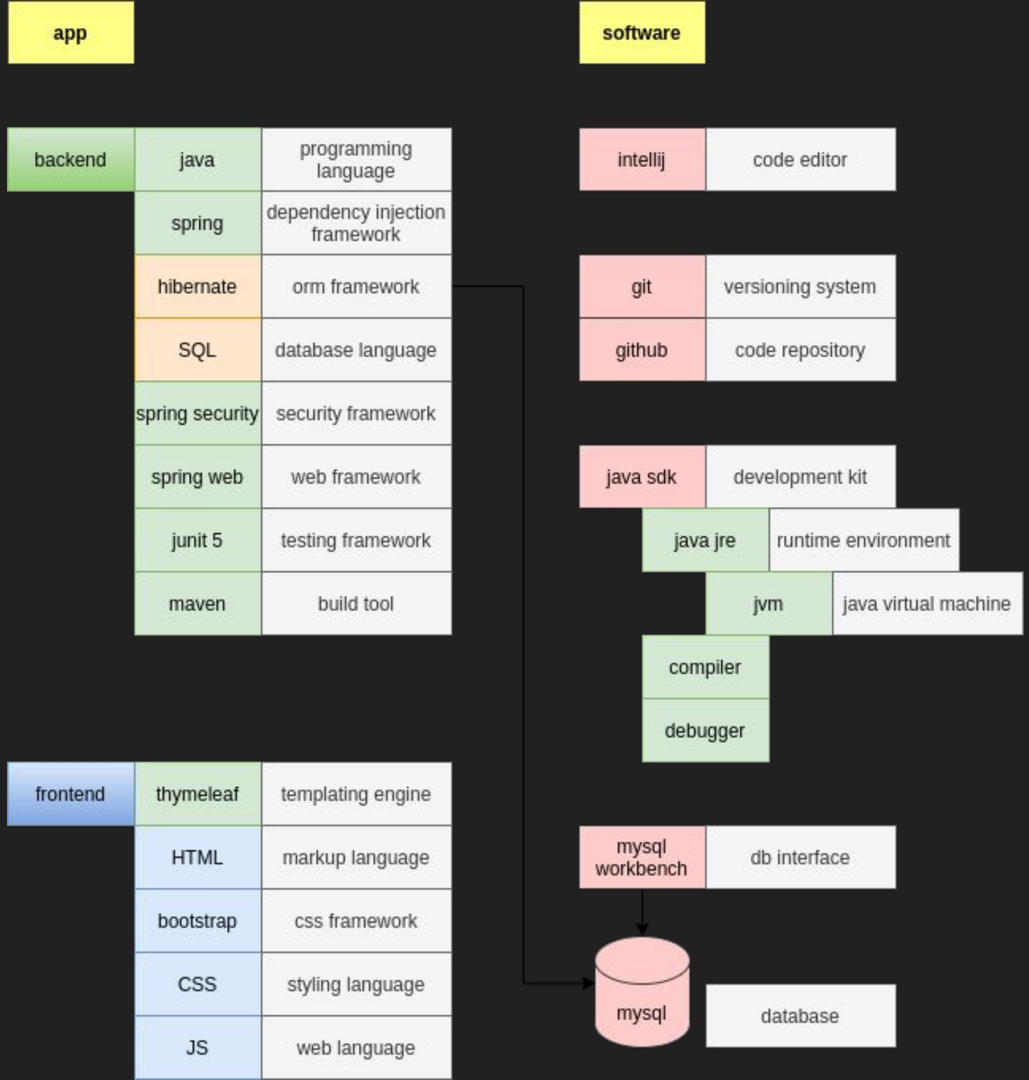
Develop

Infrastructure

steps

- Select technology stack
- Define architecture
- Define database model

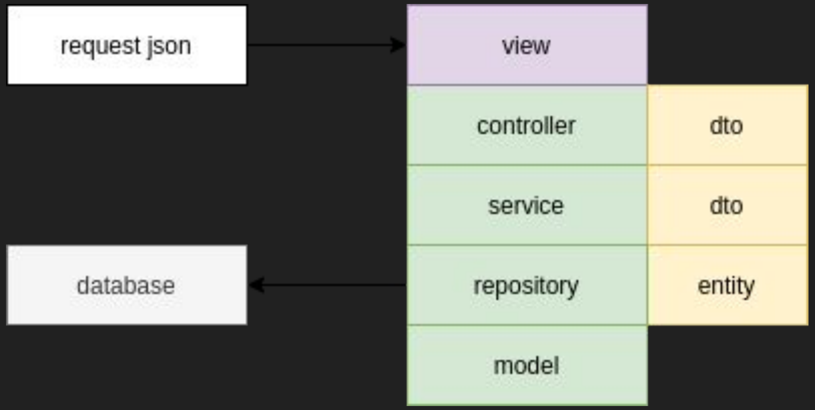
Tech stack



app		
backend	java	programming language
	spring	dependency injection framework
	hibernate	orm framework
	SQL	database language
	spring security	security framework
	spring web	web framework
	junit 5	testing framework
	maven	build tool
frontend	thymeleaf	templating engine
	HTML	markup language
	bootstrap	css framework
	CSS	styling language
	JS	web language

software	
intellij	code editor
git	versioning system
github	code repository
java sdk	development kit
java jre	runtime environment
jvm	java virtual machine
compiler	
debugger	
mysql workbench	db interface
mysql	database

Architecture



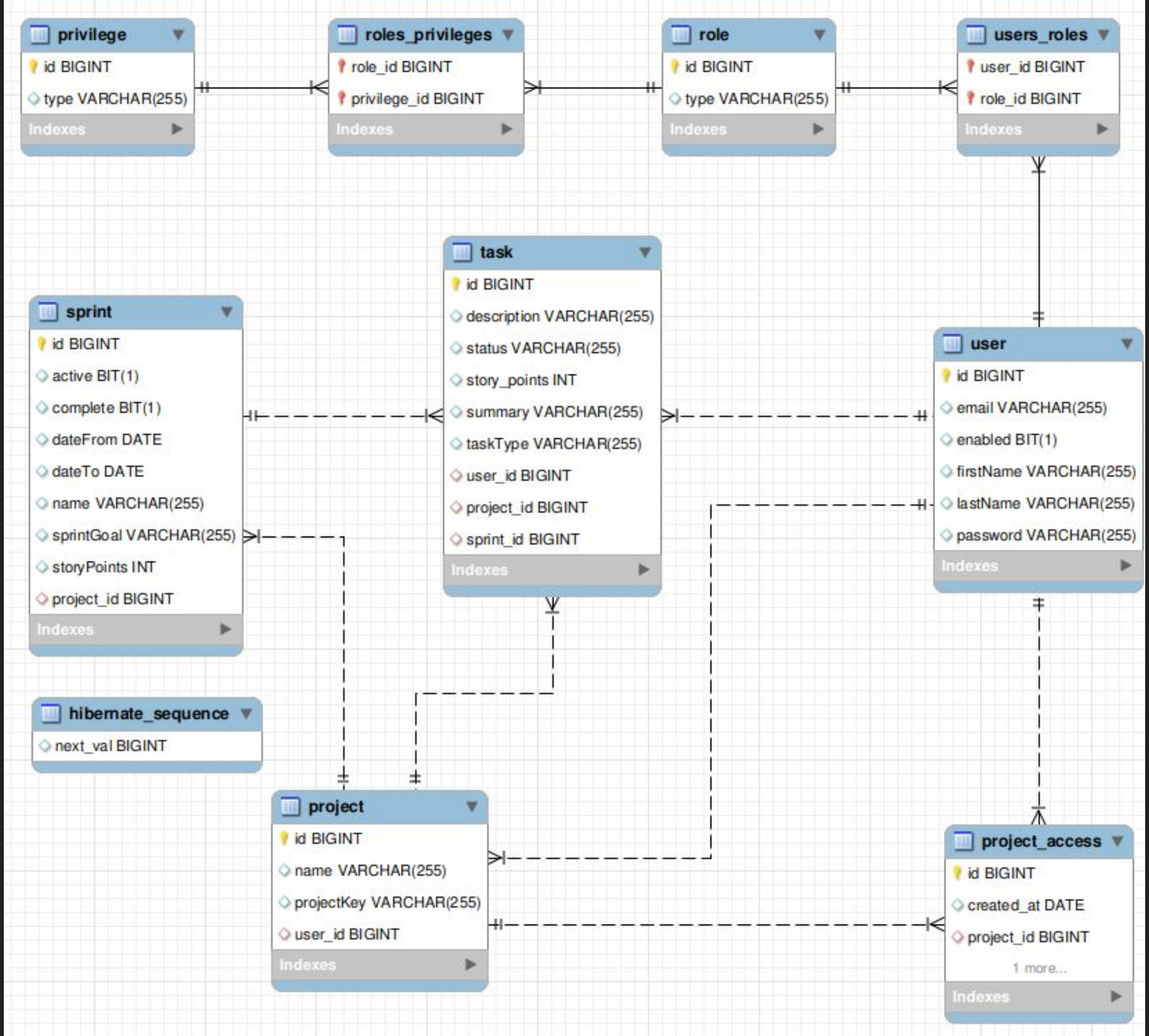
Database

Persisted the data in a MySQL database

Database

Used MySql Workbench to

- create database model
- interact with the database


privilege
id BIGINT
type VARCHAR(255)
Indexes
roles_privileges
role_id BIGINT
privilege_id BIGINT
Indexes
role
id BIGINT
type VARCHAR(255)
Indexes
users_roles
user_id BIGINT
role_id BIGINT
Indexes
task
id BIGINT
description VARCHAR(255)
status VARCHAR(255)
story_points INT
summary VARCHAR(255)
taskType VARCHAR(255)
user_id BIGINT
project_id BIGINT
sprint_id BIGINT
Indexes
sprint
id BIGINT
active BIT(1)
complete BIT(1)
dateFrom DATE
dateTo DATE
name VARCHAR(255)
sprintGoal VARCHAR(255)
storyPoints INT
project_id BIGINT
Indexes
user
id BIGINT
email VARCHAR(255)
enabled BIT(1)
firstName VARCHAR(255)
lastName VARCHAR(255)
password VARCHAR(255)
Indexes
hibernate_sequence
next_val BIGINT
project
id BIGINT
name VARCHAR(255)
projectKey VARCHAR(255)
user_id BIGINT
Indexes
project_access
id BIGINT
created_at DATE
project_id BIGINT
1 more...
Indexes

Development

steps

- Define code conventions
- Create git repository
- Create project structure
- Write tests
- Implement backend
- Implement frontend
- Add styling

Java conventions

Used java conventions and best practices

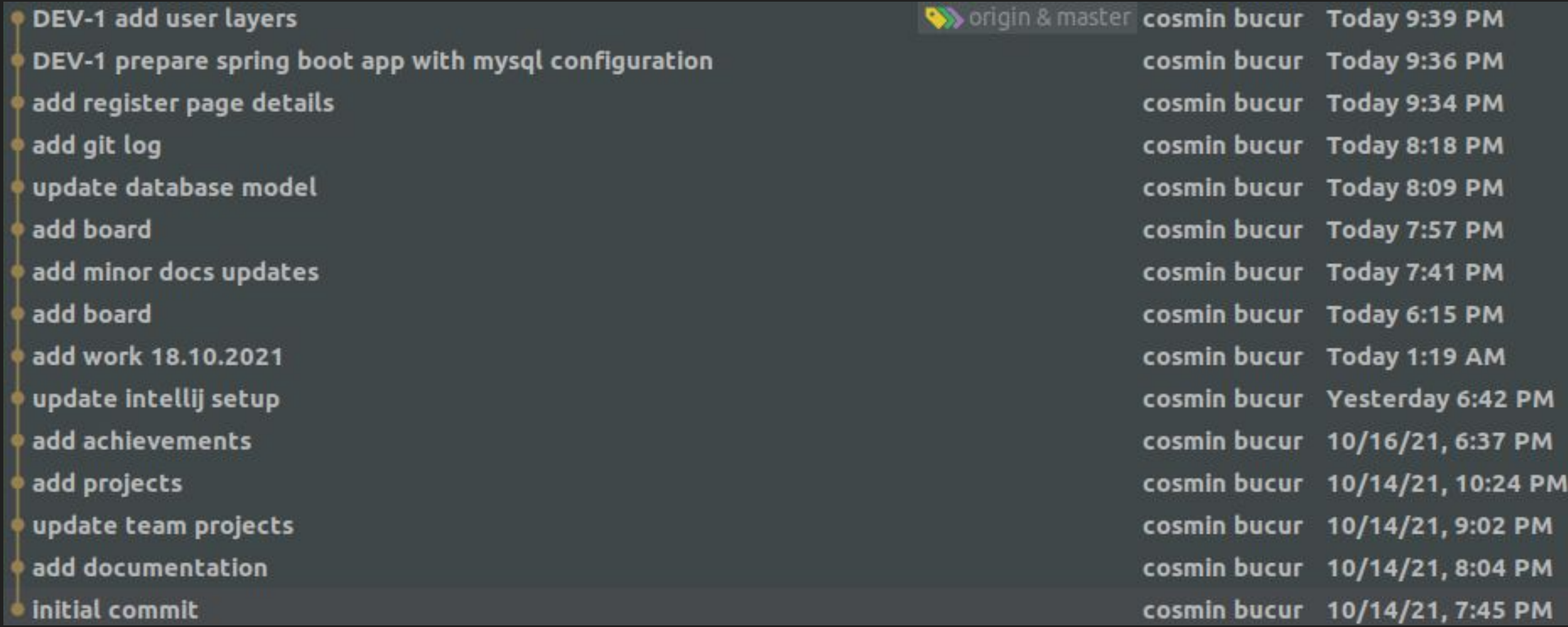
Thymeleaf conventions

Used thymeleaf conventions and best practices

Version Control System

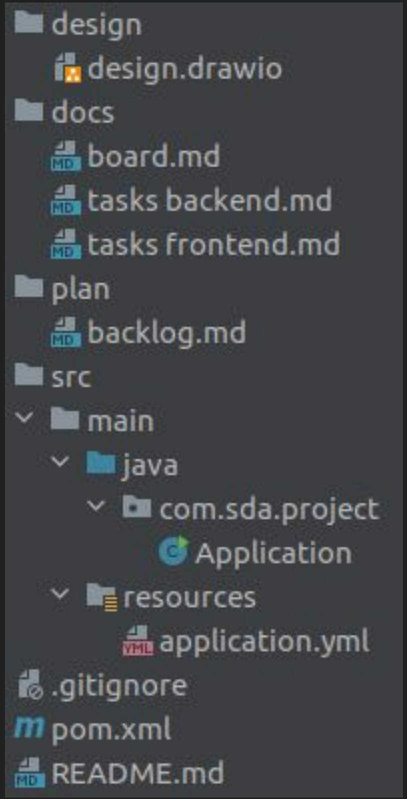
Versioned the code in a local repository using Git

Git log conventions



```
DEV-1 add user layers                                    origin & master  cosmin bucur   Today 9:39 PM
DEV-1 prepare spring boot app with mysql configuration                    cosmin bucur   Today 9:36 PM
add register page details                                                 cosmin bucur   Today 9:34 PM
add git log                                                               cosmin bucur   Today 8:18 PM
update database model                                                     cosmin bucur   Today 8:09 PM
add board                                                                 cosmin bucur   Today 7:57 PM
add minor docs updates                                                    cosmin bucur   Today 7:41 PM
add board                                                                 cosmin bucur   Today 6:15 PM
add work 18.10.2021                                                       cosmin bucur   Today 1:19 AM
update intellij setup                                                     cosmin bucur   Yesterday 6:42 PM
add achievements                                                          cosmin bucur   10/16/21, 6:37 PM
add projects                                                              cosmin bucur   10/14/21, 10:24 PM
update team projects                                                      cosmin bucur   10/14/21, 9:02 PM
add documentation                                                         cosmin bucur   10/14/21, 8:04 PM
initial commit                                                            cosmin bucur   10/14/21, 7:45 PM
```

Project structure



```
design
    design.drawio
docs
    board.md
    tasks backend.md
    tasks frontend.md
plan
    backlog.md
src
  main
      java
          com.sda.project
              Application
      resources
          application.yml
.gitignore
pom.xml
README.md
```

Code repository

Pushed the code to a GitHub repository

https://github.com/cosminbucur/project-management

Build tool

Used Maven to handle dependencies and build process

Backend

server

- Spring Boot
- Spring Data Jpa
- Spring Web
- Spring Security
- Spring Test

Spring

boot

- Service
    - Business logic
    - DTO mapper
    - Backend validation

Spring Boot

Used java 11 syntax

Used application.yml to configure spring boot

Used spring profiles to separate concerns: dev, test

Used in service layer

- business logic
- DTO mapper
- backend validation (@Valid)

Used CommandLineRunner to load initial data

Spring

data jpa

- Repository
  - Derived queries
  - Custom queries

Spring Data JPA

Used a config for each profile

- persistence JPA config
- h2 test profile config

Used Spring Data JPA repositories

- crud repository
- paging and sorting repository
- jpa repository

Used queries

- derived queries
- custom queries (HQL)

Used optional in repository methods

Used relationships

- one many
- many to many

Used Jakarta annotations

Spring

web

- Controller

Spring Web

Used ResponseEntity with DTOs

Used a Global Exception Handler to translate exceptions for user

Use web annotations

- @Controller @RestController
- @PathVariable @RequestParam
- @GetMapping @PostMapping

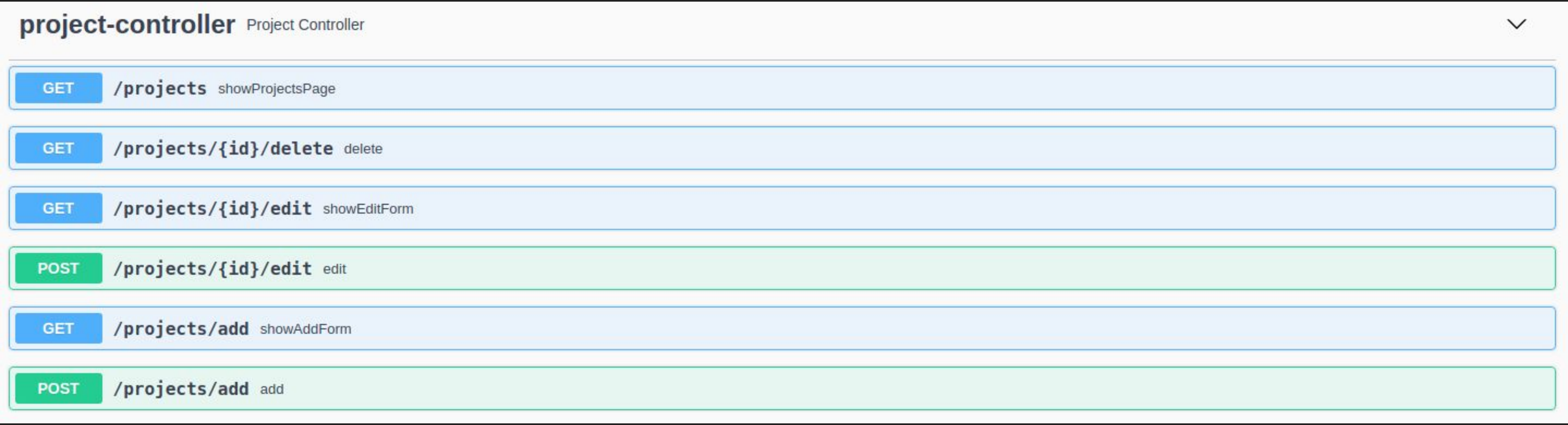
REST API convention

POST	/{collection} (json body)
GET	/{collection}
GET	/{collection}/{id}
PUT	/{collection}/{id} (json body)
DELETE	/{collection}/{id}

API documentation

Used Swagger to document the REST API

Swagger



project-controller Project Controller

GET /projects showProjectsPage

GET /projects/{id}/delete delete

GET /projects/{id}/edit showEditForm

POST /projects/{id}/edit edit

GET /projects/add showAddForm

POST /projects/add add

Logging

Used Logback to log to the console and an external file

LOGBack™
The Generic, Reliable
Fast & Flexible
Logging Framework

Spring

security

- Authentication
- Authorization
- Static resources
- User details service
- Global exception handler

Spring security

Used form authentication with a custom login page

Persisted users using user details service

Defined roles as USER, ADMIN

Spring

test

- Acceptance test
- Unit test
- Integration test

Spring test

Used TDD approach

Started from an acceptance test

Tested the service layer with

- integration tests Spring Boot Test
- unit tests with Mockito

Frontend

client

- Spring Thymeleaf
- Bootstrap
- Font awesome
- HTML
- CSS
- Java Script

Frontend stack

Increased development speed using Bootstrap components

Markup language HTML

CSS Framework Bootstrap

Styling language CSS

CSS selector jQuery

Web language JavaScript

Templating Engine

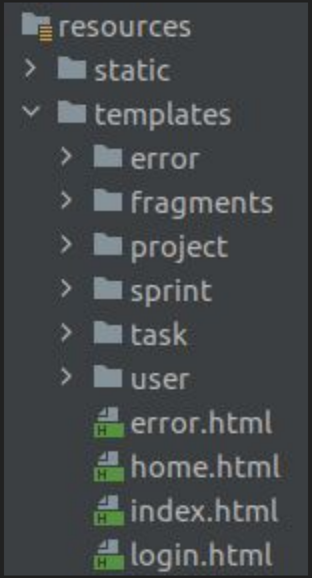
Created templates using Thymeleaf

Fragments

thymeleaf

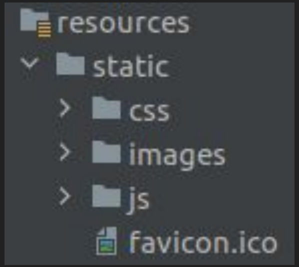
- head css
- footer js
- navbar
- sidebar

Templates



```
resources
  > static
  v templates
      > error
      > fragments
      > project
      > sprint
      > task
      > user
        error.html
        home.html
        index.html
        login.html
```

Static content



```
resources
  static
    css
    images
    js
    favicon.ico
```

Demo

MVP

- onboarding
- projects
- sprints
- tasks
- backlog
- board
- users

Fin