Veebirakenduse projekt (ITI0302)

Milestone 1

Siim Rebane

Milestone 1

- Deadline: 8th week Sunday 23:59
- 10 points, team work: one submission per team
- Written feedback, no defense
- Using LLMs is allowed
  - You have to understand everything your team submits
  - Questions about any part of the project, including a teammate's part, can be asked in the final defense

Your team

- Up to 3 students; working alone is allowed, and "per member" below then means one
- **Register your team in Moodle.** Milestone 1 is submitted by a registered team, so the team has to exist before the deadline
- The team's application runs on **one member's server**: the server issued to you in week 2
- Everything from assignment 1 about git and the pipeline applies to the team's repositories, from the first commit

What is checked, in one list

1. Topic and plan: the topic, and at least 5 GitLab issues
2. Backend: endpoints backed by PostgreSQL, in layers, with DTOs, MapStruct, Lombok, Liquibase, and the rules for the code
3. Frontend: at least one working view per member
4. Pipeline and server: every push to `main`, backend and frontend, is built, tested and deployed to the team's server
5. Git: feature branches, merge requests, a green pipeline and a teammate's review before every merge
6. Documentation: README and a wiki page

1. Topic and plan

- The project topic is part of the submission
- The topic is a web application that:
  - is backend heavy
  - uses a database
  - has some form of login
  - has some form of integration with another system
- At least **5 open GitLab issues** (or a wiki page listing them) that describe work not yet done
  - They show where the project is going after this milestone

2. Backend: endpoints

- The backend application is in git and runs
- At least **2 endpoints per team member**; each member writes their own two
- The endpoints make sense for the application; hello world does not count
- REST conventions:
  - URLs name resources, in plural (`/api/employees/5` , not `/api/getEmployee?id=5`)
  - The HTTP method names the action: `GET` , `POST` , `PUT` , `DELETE`
- Specific `2xx` status codes:
  - `POST` that creates → **201 Created**
  - `DELETE` with no response body → **204 No Content**
- Handling of `400` and `500` is not yet required

2. Backend: structure

- **Layered architecture**: controller → service → repository; a controller never calls a repository
- **DTOs** in controllers: no entity reaches the controller layer, in either direction
- **MapStruct** maps between entities and DTOs
- **Lombok** for getters, setters and constructors
- **JPA** (Spring Data repositories) for every database call
- The database is **PostgreSQL**
- The schema is defined **only in Liquibase** changesets
  - Every table has a primary key
  - Every relation between tables has a foreign key constraint

2. Backend: rules for the code

Mandatory; each one is explained in lecture 4.

- **No hand-written getters, setters or constructors**
  - DTOs are **records**
  - Entities use Lombok: `@Getter`, `@Setter`, `@NoArgsConstructor`; **no `@Data` on entities**
  - Beans get their dependencies through the constructor: `final` fields and `@RequiredArgsConstructor`; no `@Autowired` on fields
- `application.properties` sets `spring.jpa.hibernate.ddl-auto=validate` and `spring.jpa.open-in-view=false`
- PostgreSQL **18**
- Service methods that write are `@Transactional`
- A changeset that is in `main` is never edited; every schema change is a new changeset, in the same commit as the entity change

3. Frontend

- The frontend application is in git and runs
- Vue, React or Angular
- At least **1 working view per team member**: an input form, a data view, anything that uses a backend endpoint
- Each member develops at least one feature
- A developed feature must work; the whole flow does not have to yet
  - For example: the view works, but the menu link to it is not there yet
- Functionality that is hard to find in the interface is described in the submission

4. Pipeline and server: backend

Mandatory, as in assignment 1:

- A `.gitlab-ci.yml` with **build, dockerize, deploy**; dockerize and deploy only on `main`
- The build job runs the tests, with a **Postgres service** so they can start the application (lecture 4)
- The deploy job puts the backend **and PostgreSQL** on the team's server with Compose, the data in a named volume
- The database password is a masked CI/CD variable, never in git
- `GET /api/version` on the server returns the commit it was built from
- A push to `main` changes what runs on the server, with no step done by hand

4. Pipeline and server: frontend

Mandatory, the same way as the backend:

- Backend and frontend may be two repositories, or two folders in one repository; either way the frontend has its own **build** and **deploy** jobs, with deploy only on `main`
- The build job builds the production version of the frontend
- The deploy job copies the built files onto the **same team server** as the backend, into its nginx (`/var/www/html`, lecture 6)
- `http://<server>/` opens the application, and its requests to `/api` reach the backend
- A push to `main` changes what runs on the server, with no step done by hand

5. Git

Mandatory, as in assignment 1:

- All work goes through **feature branches and merge requests**; nobody pushes to `main`
- A merge request is merged only with a **green pipeline**
- In a team of two or three, a **teammate reviews and approves** the merge request before it is merged
- `main` is a protected branch: merging only through merge requests

The history is read: who wrote what, which branch, which merge request, who approved it.

6. Documentation

README in the repository:

- how to build the application
- how to run it, including the database

A wiki page in GitLab:

- the project idea: more than one paragraph, with a clearly described end result
- mockups or sketches of the interface are a big plus
- the technologies used
- the first version of the database, **with a database diagram**

Deliverable

- In Moodle, one submission per registered team:
  - the topic
  - the link to the repository, or both links if backend and frontend are separate
  - the address where the application opens on the team's server
  - functionality that is hard to find in the interface, and where it is
- Every repository's name starts with `iti0302-2026`, for example `iti0302-2026-<team>-backend`; that is what gives the lecturer access
- What is checked is the state at the deadline: the repositories, both pipelines, and the application on the server



Questions