

# PROJEKTARBETE | Complex Java | JU20 | ITHS

## Gemensam Rapport

Albert Andersson, Casper Konyi, Jannis Müller, Joakim Lidén

- Presentera projektet

Vi har skapat en chat-applikation. Målet var att skapa en plattform där man kan registrera sig och chatta med andra användare i olika kanaler (feeds). Administratörer ska ha vissa rättigheter, som t.ex. att skapa en ny admin-användare och se alla registrerade användare.

Vi har använt oss utav Spring som backend, MariaDB som databas och Thymeleaf med Bootstrap som frontend.

- Beskriva arbetsprocessen

Vi började med att skapa en organisation i Github och sätta upp ett repository. Första steget var att sätta upp ett Spring-projekt i IntelliJ där vi skapade en simpel grundstruktur (entiteter, repositories, controllers, databaskonfiguration, etc) som fungerade på alla gruppmedlemmars enheter.

I nästa steg gjorde vi en kanban-tavla med kolumnerna `Todo`, `In progress` och `Done`. Efter det så började vi skapa tickets som fyllde på vår `Todo`-kolumn. Varje ticket var kopplat till en `Issue` som man referade till när man branchade ut. Vi prioriterade vår `Todo` gemensamt och sedan delade vi ut arbetsuppgifter.

Vi höll kontinuerlig kontakt över Teams och hjälptes åt när det behövdes, vissa tickets gjordes helt själv och andra löste vi med parprogrammering/mobb-programmering. Arbetsflödet var att skapa en ny branch för varje ticket och kontinuerligt rebase sin branch från master. När man var klar gjordes en pull-request mot master och man meddelade gruppen. Vi satte tidigt upp en Github Action körde våra tester när man gjorde en pull-request mot master - gick in testerna igenom så fick man inte mergea in. Detta arbetsflöde underlättade mycket för oss då det höll master i ett bra state och gjorde det enkelt att arbeta individuellt.

- Tekniska val och implementation

Vi valde att använda oss av enkel säkerhet och inte köra med JWT. Varje lösenord körs genom BCrypt innan det sparas ner. Huvudsyftet var att endast inloggade användare ska kunna använda applikationen. Vissa endpoints var endast tillgängliga för användare med rollen `ROLE_ADMIN`.

För att slippa mycket boilerplate-kod så använde vi oss av biblioteket Lombok, som t.ex. `@Getter`, `@Setter` i våra entiteter.

Mapping mellan DTO - Entitet hjälpte MapStruct till med, nedan är ett exempel för att mappa `UserEntity` - `UserDTO`.

```
@Mapper(componentModel = "spring", uses = {RoleMapper.class})
public interface UserMapper {

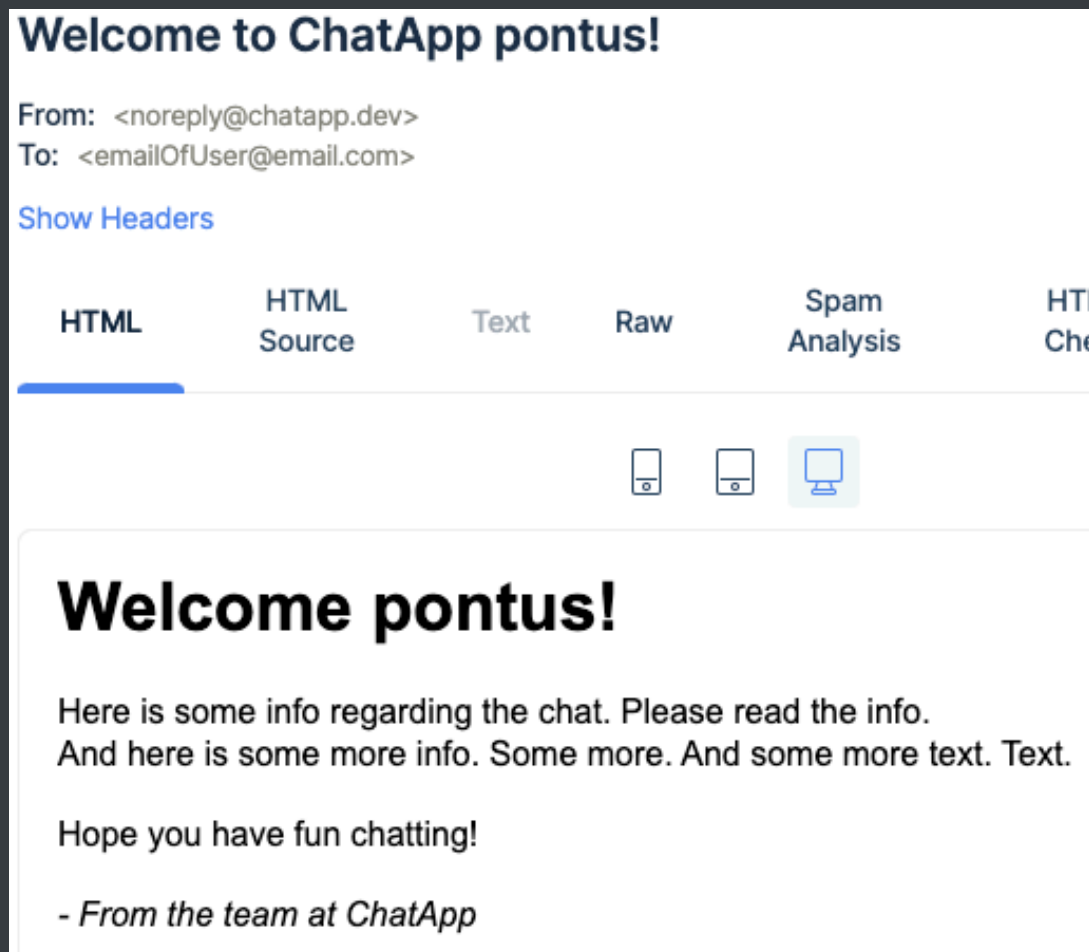
    @Autowired
    UserMapper INSTANCE = Mappers.getMapper(UserMapper.class);

    @Mapping(target = "messages", ignore = true)
    User toDto(UserEntity entity);

    @Mapping(target = "messages", ignore = true)
    UserEntity fromDto(User user);

}
```

När en användare registrerar sig får den ett



- Lombok
- Mapstruct
- JMS (Artemis)
- Spring Mail
- Docker, Docker Compose
- Github kanban, issues, pull request per ticket
- CI / Webhooks
  - Testrapport
    - Skydd av master
  - Skapande av artefakt

▪

