Coding Guidelines

Generelt

- Det anbefales at lægge lokal kode under C:\dev\source
- Kode, kommentarer og commit beskrivelser skal som udgangspunkt altid være på engelsk
    - Undtaget er forretningsbegreber, der kan skabe forvirring hvis de oversættes direkte (f.eks. "Anmodning" i DMR)
- Det er vigtigere med kode der nemt kan læses, end "korte one liners" (se i øvrigt Clean Code guidelines)
- Husk at skrive kommentarer - både Javadoc og som hjælp til læsning i selve koden
- Husk at opdatere kommentarer og Javadoc ved ændring af eksisterende kode
- En god kommentar beskriver hvad og hvorfor noget sker, ikke hvordan
- Husk at skrive/opdatere readme og anden dokumentation, samt diagrammer - se Diagrams as Code
- Der skrives automatiseret test til al ny kode (hvor det giver mening) - se Automated tests
- Gør dig bekendt med vores politik for brug af AI

Git

- Udfyld navn i Git config
- Branches følger Git flow - se Git flow
- Branch merge sker i udgangspunktet altid med code review - se Code review
- Commit beskrivelser skal altid indeholde et Jira sagsnr. (tænk sporing fra kodeændring til opgavebeskrivelse)
- Versionsopdatering sker i develop efter release, eller i release og hotfix branches efter behov - se Versionering

Opdel opgaver

- Hvis opgaver er for store, så opdel dem i mindre bidder, og lav pull request for hver bid - det giver et hurtigere feedback loop
- Hvis scope eller opgaven ændrer sig undervejs og bliver større, så opdel den i mindre logiske bidder - opret relevante Jira tasks
- Opdagede scope ændringer vendes altid med projektlederen
- Hvis en opdelt bid ikke meningsfyldt kan merges ind i develop branch, så håndtér branches med en “feature main branch” og “sub feature branches”

Specifikke cases

Java / Spring

- Se Formatering af Java kode

- Apache Commons library bruges som hjælpemetoder
    - Brug static imports for at gøre det nemmere at læse
- Lombok library bruges for at undgå for meget boiler code
- Pas på med generiske catch-all serviceklasser, der får for meget ansvar. Opdel det i stedet i flere, mindre serviceklasser med udgangspunkt i use cases.
- Ved brug af streams, indsæt linjeskifte efter hvert metodekald, for at gøre det nemmere at læse
- Undgå inline servicekald, da det gør det sværere at læse og fejlsøge. Et inline servicekald er kald af metode i serviceklasse, indeni et andet statement, f.eks. en setter.
- Se Spring konfigurationsfiler
- Se Navngivning af Maven moduler
- Se Libraries

Frontend

- See Getting Started with Frontend

Yderligere læsning

- Clean Code guidelines
- Spring Boot Microservices Coding Style Guidelines and Best Practices
- Zalando RESTful API and Event Guidelines
- Top 20 Java Exception Handling Best Practices