

Mikkel skal lave sine fod noter færdig, i metode afsnittede

# Indholdsfortegnelse

Indholdsfortegnelse.....	2
Indledning.....	6
Bankdata .....	6
Opgaveformulering.....	6
Projektets formål og en kort beskrivelse .....	6
Problemstilling .....	7
Følgende problemstillinger ønskes behandlet. ....	7
Krav til løsningen .....	7
Alternativ til Bankdata's CI/CD -pipeline .....	7
Overordnet hvilke arbejdsopgaver jeg vil arbejde med, evt. en foreløbig løsningsbeskrivelse.....	8
Metoder .....	8
UML .....	8
User stories.....	8
Use case .....	8
Use case diagram.....	9
Domain model.....	9
System sekvens diagram.....	9
Operations kontrakt.....	9
Sekvens diagram .....	9
Design class diagram .....	10
Database model.....	10
Scrum.....	10
Kanban.....	10
Agile .....	10
Teknologier og værktøjer .....	10
Teknologier .....	11
Kubernetes – Kind .....	11
Java v.17 .....	11
Angular.....	11
JPA – Hibernate ORM .....	11
Quarkus.....	12
OpenAPI.....	12

Værktøjer .....	12
Visual Studio Code .....	12
Jira.....	12
Confluence .....	12
Lucid shardt.....	13
GitHub .....	13
Codespace .....	13
Github actions .....	13
Docker .....	13
DockerHub .....	13
Projektetablering .....	13
Aftale med Bankdata .....	14
Valg af project form og arbejdsmetoder .....	15
Agilt .....	15
Scrum Kanban .....	15
Sprint planning .....	15
Roadmap .....	15
Dayli scrum meeting .....	16
Sprint review .....	16
Kanban-bræt .....	16
Valg og opsætning af project managements tools .....	16
Jira .....	16
Confurence .....	16
Valg af udvikling sprog og tool change .....	20
Udvikling sprog og frameworks .....	20
Github .....	20
Visual Studio Code .....	20
System .....	20
Analyse.....	21
User stories.....	21
Use cases .....	21
Use Case Diagrams (Behavioral).....	22
Design .....	23
Component Diagram (Structural) .....	23
Class Diagrams (Structural) .....	24
Package Diagrams (Structural).....	24

Lagdelt systemdesign.....	25
Object Diagrams (Structural).....	25
Interaction Diagram (Behavioral) .....	25
Livscyklus for deployment.....	25
Sequence Diagram (Behavioral) .....	26
Implementering.....	26
Hvordan få man adgang til coden og codespaces.....	26
Hvordan tester man det.....	26
Hvordan start man programmet .....	27
Backend Rest API .....	29
Rest Api.....	29
Rest Client kald .....	30
JPA – Hibernate .....	31
Interface .....	31
Metriker .....	32
Applikation .....	32
Kubernetes .....	33
Frontend UI.....	34
GitHub Codespace.....	34
Bankdatas situation i dag .....	34
FAK konceptet .....	34
Låst maskine.....	35
Inrollede på Bankdata network.....	35
Genetablering af maskiner .....	35
Hvad er GitHub codespace.....	35
Dimensionering af codespace .....	37
Custom konfiguration af codespace .....	37
Rebuild .....	37
Port forwarding .....	38
Design og implementering af codespaces.....	38
Codespace til Frontend udviklere.....	39
Codespace til Backend udviklere .....	40
Codespace til integrations / verifikations miljø .....	41
Implementering af codespaces .....	41
Installe-services start script .....	42
CI / CD.....	43

GitHub Actions (CI) .....	43
Hvad er GitHub Actions .....	43
Design og implementering af Actions .....	43
Java Backend build pipeline (pipeLine.yml) .....	43
Angular Frontend build pipeline .....	44
ArgoCd (CD).....	44
Hvad er ArgoCd .....	44
Design og implementering af ArgoCd .....	44
Konklusion.....	45
Codespace .....	45
Systemudvikling .....	45
Project management .....	45
Samarbejde med Bankdata .....	45
Kildehenvisninger.....	46
Links .....	46
Bøger .....	48
Bilag .....	48
Test bruger .....	48

# Indledning

I denne indledning vil vi gennemgå de beslutninger og prioriteringer, der er blevet truffet med udgangspunkt i opgaveformuleringen, og som har været grundlaget for vores arbejde med projektet.

På grund af omfanget af opgaveformuleringen var der allerede under godkendelsesprocessen en debat om, at den var for ambitiøs. Bankdata ved Tom Ingemand Nielsen<sup>1</sup> udtrykte under denne debat, at de hellere ville være ambitiøse og så ikke nå helt i mål.

På grund af denne beslutning har der været behov for at lave prioriteringer sammen med Bankdata.

Prioriteringer er blevet lavet pr. problemstilling, og det er blevet besluttet, at de to problemstillinger skal vægtes lige højt. Derefter er der lavet en prioriteret liste for hver problemstilling, og listen er i prioriteret rækkefølge. Den prioriteret liste fremgår under projektetablering.

## Bankdata

Bankdata er en dansk IT-virksomhed, der leverer it-løsninger og serviceydelser til finanssektoren. De tilbyder en række forskellige produkter og tjenester, herunder it-infrastruktur, it-sikkerhed, it-drift og it-udvikling. Bankdata har også et fokus på digitalisering og agile metoder, og de arbejder på at skabe en mere smidig og effektiv it-drift for deres kunder. Du kan finde yderligere oplysninger om Bankdata og deres forretningsområder på deres hjemmeside.<sup>2</sup>

## Opgaveformulering

### Projektets formål og en kort beskrivelse

Bankdata har undergået en rejse mod en devops drevet organisation, det vil sige at udvikler er blevet mere og mere vandt til at bygge og deployer deres artefakter.

Formålet med opgaven og sammenarbejdet med Bankdata, er at styrke deres rejse mod et devops drevet organisation. Under udarbejdelsen af opgaven vil jeg udvikle et produkt der vil kunne give overblik over hvilke artefakter der er deployet, hvilke platforme og miljø det er ske til.

Det færdige produkt vil kunne frem vise de overstående punkter i en webapplikation. I webapplikationen kommer der til at være mulighed for at kunne søge og filtrere i det forskellige deployments. Dataene der vil blive vist i webapplikationen. vil komme kald til micro-services der vil blive udviklet i Java med et framework der hedder Quarkus og skal køre som en container løsning på kubernetes.

---

<sup>1</sup>Tom Ingeman Nielsen Afdelingsleder Decentrale Udviklingsplatforme

<sup>2</sup> Bankdata hjemmeside: [www.bankdata.dk](http://www.bankdata.dk)

# Problemstilling

## Følgende problemstillinger ønskes behandlet.

- System til registrering af hvor artifact er deployed
- Alternativ til Bankdata's CI/CD -pipelines

### System til registrering af hvor artifact er deployed

Bankdata er på en rejse mod en DevOps præget organisation, det er i denne proces blevet mere og mere almindeligt at udviklings teams selv bygger og deployer deres artefakter. Deployment blev tidligere varetaget af centrale funktioner som f.eks. Platform teamet, som derfor kontrollerede og styre hvilke artefakter blev deployet og hvor til de blev deployet. Denne viden er central for platformsteamet i forbindelse med etablering af nye varianter af en Platform, f.eks i forbindelse med etablering af konverterings miljøer til Medarbejder og Netbank Portalerne, eller i forbindelse med en disaster recovery, hvor man er tvunget til at gen deploye alle artefakter.

Bankdata ønsker derfor at udvikle et centralt system hvor teams kan registrere, hvilke artefakter de deployer på hvilke platforme og miljøer.

### Krav til løsningen

- Web interface til søgning
  - Hvor er en given artefakt deployet
  - Hvilke artefakter er deployet på en given platform og miljø
- Dataintegritet skal sikres
- Konfiguration af systemet skal være som source i source repositories
- Monitoring
  - Forbrugs statistiker
  - Systemets ressourceforbrug
- CI/CD pipe line
- Afvikling platformen skal være Kubernetes

### Alternativ til Bankdata's CI/CD -pipeline

Bankdata rejse mod en devops præget organisation, gør også at Bankdata gerne vil gøre det lettere at udvikle, bygge, teste og deploy.

Bankdata's CI/CD -pipelines basere sig i dag på mange komponenter fra forskellige leverandører og det kan være en besværlig opgave at sætte en byg og test op på en lokalt Bankdata maskine, da den rettighedsmæssigt er meget låst og g bag en proxy.

Så Bankdata ønsker eksempel på hvordan man kan lave CI/CD -pipeline, som er mere cloude enlablede så man ikke er så afhængig de lokale maskiner, dette kunne gøre ved at lave en CI/CD -pipeline til systemet der lave til at løse problemstillingen 'System til registrering af hvor artefakter er deployed'

## Overordnet hvilke arbejdsopgaver jeg vil arbejde med, evt. en foreløbig løsningsbeskrivelse

Det overordnede arbejdsopgave er at finde en den bedste teoretiske løsning på bankdatas problem, og samme tid holde en tæt kontakt med dem, for at sikre produktet passer til deres krav og efterspørgelse. derefter udarbejde en løsning i form af applikationer.

Løsnings på de for skellige problemstillinger og krav, vil blive løst ved brug af følgende frameworks og tools.

- Micro-servicerne vil blive udvikle i Quarkus, som er et java baseret framework.
- Webapplikationerne vil blive udarbejde i Angular. Angular er en komponent baseret framework til udvikling af skalerbare webapplikationer.
- Data bliver lagret i en MariaDB.
- Montering af system vil blive gjort med Prometheus og vist i Grafana dashboard.
- CI/CD-pipelines udvikles med GitHub Actions, GitHub CodeSpace og ArgoCD
- Kubernetes til afviklings platform.

## Metoder

I det følgende afsnit vil der blive givet en gennemgang af de metoder, der er blevet anvendt i projektet. Dette vil inkludere en oversigt over de teknikker og værktøjer, der er blevet benyttet, samt en beskrivelse af metodologien og de processer, der er blevet fulgt for at opnå projektmålene. Derudover vil der også blive beskrevet, hvad metoder, teknikker og værktøjer kan gøre og hvad de er designet til at opnå.

## UML

### User stories

En user story er en kort beskrivelse af en funktionalitet eller et forretningsbehov, som er formuleret fra brugerens perspektiv. Det er en måde at kommunikere krav og ønsker fra brugere til udviklere på, ofte i forbindelse med agile softwareudviklingsmetoder.<sup>3</sup>

User stories er skrevet i samarbejde med Bankdata

### Use case

En use case beskriver en række trin, som systemet skal gennemføre for at opfylde brugerens behov. Hvert trin i en use case kaldes en "scenario", og hvert scenario beskriver en handling, som systemet skal udføre, og et resultat, som brugeren forventer at se. Use cases kan også indeholde alternative scenarier, der beskriver hvad der sker, hvis der opstår problemer eller uventede hændelser under interaktionen.<sup>4</sup>

---

<sup>3</sup> User stories: <https://www.agilealliance.org/glossary/user-stories>

<sup>4</sup> use cases: <https://www.usability.gov/how-to-and-tools/methods/use-cases.html>



## Use case diagram

Et use case diagram er et diagram, der visualiserer interaktionerne mellem aktører og et system i en given situation. Det består af aktører, der er personer eller andre systemer, der interagerer med systemet, use cases, der er de specifikke handlinger eller aktiviteter, som systemet udfører, og forbindelser, der er linjer der viser hvordan aktørerne interagerer med use cases<sup>5</sup>

## Domain model

Domain modeller er nyttige, fordi de gør det muligt at strukturere og organisere viden om et domæne på en overskuelig måde. De er også nyttige til at identificere og definere de konceptuelle elementer inden for et domæne og de relationer, der findes mellem dem. Dette kan hjælpe med at klarlægge forståelsen af domænet og sikre, at alle relevante aspekter bliver taget i betragtning under udviklingen af et system eller en løsning.<sup>6</sup>

## System sekvens diagram

Et system sekvens diagram (SSD) er et diagram, der viser trin eller sekvenser, der udføres i et system for at opfylde et specifikt behov eller udføre en specifik aktivitet. Det består af aktører, der er personer eller systemer, der interagerer med systemet, sekvenser, der er de specifikke trin eller handlinger, som systemet udfører, objekter, der er de ting, som systemet interagerer med eller manipulerer, og systemet selv, der vises som en ramme omkring sekvenserne<sup>7</sup>

## Operations kontrakt

En operationskontrakt er en aftale, der beskriver hvordan et system skal fungere og hvad der forventes af systemet i en given situation. Den består af input, der er de data eller parametre, som systemet modtager, output, der er det resultat, som systemet forventes at producere, pre- og postkonditioner, der er krav til systemet før og efter en given operation, og fejlhåndtering, der beskriver hvordan systemet skal håndtere fejl, der måtte opstå under operationen.<sup>8</sup>

## Sekvens diagram

SD'er er nyttige til at visualisere hvordan et system opfylder et specifikt behov eller udfører en specifik aktivitet. De er også nyttige til at identificere mulige problemer eller uensigtsmæssigheder i systemet og til at planlægge hvordan systemet kan testes.<sup>9</sup>

---

<sup>5</sup> use case diagram (S. 3 –10)

<http://csis.pace.edu/~marchese/CS389/L9/Use%20Case%20Diagrams.pdf>

<sup>6</sup> APPLYING UML AND PATTERNS by CRAIG LARMAN (S. 134)

<sup>7</sup> APPLYING UML AND PATTERNS by CRAIG LARMAN (S. 176)

<sup>8</sup> APPLYING UML AND PATTERNS by CRAIG LARMAN (S. 181-183)

<sup>9</sup> APPLYING UML AND PATTERNS by CRAIG LARMAN (S. 222)

## Design class diagram

Design class diagrammer er nyttige, fordi de gør det muligt at strukturere og organisere viden om systemet på en overskuelig måde. De er også nyttige til at identificere og definere de konceptuelle elementer inden for systemet og de relationer, der findes mellem dem. Dette kan hjælpe med at klarlægge forståelsen af systemet og sikre, at alle relevante aspekter bliver taget i betragtning under udviklingen af systemet.<sup>10</sup>

## Database model

Et database model er en form for teknisk model, der bruges til at strukturere og organisere data i en database. Det beskriver hvordan dataene er struktureret og relateret til hinanden, og hvordan de kan hentes og manipuleres.<sup>11</sup>

## Scrum

Scrum er en populær metode inden for agile softwareudvikling, fordi den gør det muligt at levere værdi hurtigt og effektivt og fordi den giver teamet stor frihed til selv at organisere og planlægge sin arbejdsproces.<sup>12</sup>

## Kanban

Kanban er en metode til at styre arbejdsflow og planlægning i projekter og processer. Det fokuserer på at visualisere arbejdsflowet og at justere arbejdsbelastningen i realtid for at forbedre effektiviteten og reducere spild.

Der er lavet et flow for processer se under projekt etablering<sup>13</sup>

## Agile

Agile er en tilgang til softwareudvikling, der fokuserer på at levere værdi hurtigt og effektivt ved at være fleksibel og tilpasse sig ændringer i krav og prioriteter. Agile metoder lægger vægt på samarbejde med kunden, selvorganiserende teams og kontinuerlig levering af funktionalitet.

Iterativt gennem tæt samarbejde mellem udviklere og Bankdata<sup>14</sup>

## Teknologier og værktøjer

Her bliver Følgende en liste med teknologier og værktøjer præsenteres, samt en forklaring på, hvad de gør og hvordan de er blevet anvendt i projektet. Dette giver en forståelse af, hvilke teknologier og værktøjer der er blevet brugt i projektet, og hvad deres formål og rolle er i forhold til projektet. Dette er vigtigt, da det giver et overblik over, hvordan projektet er

---

<sup>10</sup> class diagram: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

<sup>11</sup> database design: Database concepts-Pearson (S. 261)

<sup>12</sup> Scrum: <https://www.scrum.org/resources/what-is-scrum>

<sup>13</sup> kaban: <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban>

<sup>14</sup> agile: <https://www.atlassian.com/agile>

blevet til og hvilke teknologier og værktøjer der har været med til at forme og realisere projektet.

## Teknologier

### Kubernetes – Kind

Kind er et open source-værktøj der kan køre Kubernetes clustere i docker. Der hjælper med at styre container-baserede applikationer på flere maskiner og automatisere driften af dem. Det giver mulighed for skalering, ressourcekonfliktløsning og kontinuerlig levering.<sup>15</sup>

Bruges til integrations / verifikations miljø, hvor det giver mulighed for at kontrollere at følgende krav er blevet opfyldt og virker

- Metriker via prometheus format
- Monitorering via Grafana
- CD – ArgoCD
- Afvikling platform Kubernetes

### Java v.17

Java er et populært programmeringssprog, der anvendes i mange forskellige sammenhænge, herunder webudvikling, mobile apps og backend-systemer. Det er kendt for sin portabilitet, så kode, der er skrevet i Java, kan køre på mange forskellige platforme uden ændringer. I projektet bruges som programmering ved udvikling af backend systemet.<sup>16</sup>

### Angular

Angular er et JavaScript/typescript-framework til udvikling af moderne webapplikationer. Det strukturerer applikationen som en samling af komponenter og tilbyder værktøjer til templating, dependency injection og testning. Det er populært på grund af sin evne til at håndtere store og komplekse applikationer. Angular isammen arbejde med typescript Bruges til udvikling af frontend UI i systemet, her under bruges typescript hovedsagelig til sammenkobling af backenden og frontenden.<sup>17</sup>

### JPA – Hibernate ORM

JPA (Java Persistence API) er en standard for at gemme Java-objekter i en database, ved at bruge Object/Relational Mapping (ORM) framework. Hibernate er en open source-implementering af JPA, der tilbyder værktøjer til at håndtere persistence i Java-applikationer. Hibernate gør det nemt at mappe Java-objekter i en relationel database ved hjælp af et enkel API.<sup>18</sup>

---

<sup>15</sup> Kind: <https://kind.sigs.k8s.io/>

<sup>16</sup> Open JDK 17: <https://openjdk.org/projects/jdk/17/>

<sup>17</sup> Angular: <https://angular.io/guide/what-is-angular>

<sup>18</sup> JPA – Hibernate ORM: <https://hibernate.org/orm/>

## Quarkus

Quarkus er et open source-microservices-framework, der er designet til at være så lille og effektiv som muligt. Det er bygget til at køre på en række forskellige platforme, herunder containers, Kubernetes og OpenShift. Det tilbyder en række funktioner, der hjælper med at gøre udviklingen af microservices nemmere.<sup>19</sup>

## OpenAPI

OpenAPI UI (tidligere kendt som Swagger UI) er et open source-værktøj til at visualisere og interagere med RESTful API'er, der er beskrevet ved hjælp af OpenAPI-specifikationen. Det gør det muligt at generere en interaktiv API-dokumentation ved hjælp af en specifikation i form af en YAML- eller JSON-fil, der definerer endepunkter, metoder, parametre og responser for API'en. OpenAPI UI tilbyder også en række værktøjer til at teste og interagere med API'er, såsom muligheden for at udføre forskellige HTTP-metoder, indsende parametre og se responser.<sup>20</sup>

## Værktøjer

### Visual Studio Code

Visual Studio Code er en gratis, open source-koderedigeringssoftware, der er udviklet af Microsoft. Det tilbyder en række funktioner, der er designet til at gøre udviklingen nemmere, såsom intelligent kodefærdiggørelse, debugging, testning og version kontrol og understøtter mange kode sprog i gemmen ekstensions<sup>21</sup>

### Jira

Jira er et værktøj til projektstyring og problemlapportering, der hjælper med at spore opgaver, bugrapporter og ønsker fra kunder. Det tilbyder muligheden for at oprette og spore issues, tildele opgaver, holde styr på fremskridt og samarbejde om løsninger.<sup>22</sup>

### Confluence

Confluence er et værktøj til samarbejde og dokumentation, der hjælper med at dele og samarbejde om dokumenter, ideer og projekter. Det tilbyder muligheden for at oprette og redigere dokumenter, tildele opgaver og samarbejde via kommentarer og vedhæftede filer. Confluence er designet til at være en central platform for samarbejde om projekter.<sup>23</sup>

---

<sup>19</sup> Quarkus: <https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus>

<sup>20</sup> OpenAPI: <https://spec.openapis.org/oas/v3.1.0>

<sup>21</sup> Visual Studio Code: [https://en.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)

<sup>22</sup> Jira: <https://www.productplan.com/glossary/jira/>

<sup>23</sup> Confluence: <https://www.atlassian.com/software/confluence/guides/get-started/confluence-overview#hosting-options>

## Lucid shardt

Lucidchart er et online diagramværktøj, der hjælper med at oprette og redigere forskellige typer af diagrammer. Det tilbyder skabeloner, import og eksport af filer og integrerede samarbejdsfunktioner.<sup>24</sup>

## GitHub

GitHub er et webbaseret platform til opbevaring og administration af kode, samarbejde om projekter og dele arbejde med andre. Det tilbyder versionkontrol, bugrapportering, dokumentation og samarbejdsfunktioner. GitHub er populært på grund af sin evne til at gøre samarbejde og vedligeholdelse af kode nemmere<sup>25</sup>

## Codespace

GitHub Codespaces er et online udviklingsmiljø, der hjælper med at oprette, redigere og køre kode fra skyen. Det tilbyder intelligent kodefærdiggørelse, debugging, integrerede værktøjer til versionkontrol og samarbejde. GitHub Codespaces er baseret på Visual Studio Code og understøtter en række sprog og platforme. Det er populært på grund af sin evne til at gøre udviklingen nemmere og brugervenligheden høj.<sup>26</sup>

## Github actions

GitHub Actions er en service, der hjælper med at automatisere workflows omkring projekter på GitHub. Det tilbyder muligheden for at automatisere bygge, test, deployment og andre processer og integrerer med andre tjenester og værktøjer.<sup>27</sup>

## Docker

Docker er en teknologi, der hjælper med at pakke en applikation og dets afhængigheder i en container, der kan køre på enhver platform. Docker gør det nemt at distribuere og vedligeholde applikationer og nemmere at samarbejde om projekter<sup>28</sup>

## DockerHub

Docker Hub er en tjeneste, der hjælper med at dele, opbevare og administrere containerbilleder. Det gør det nemt at samarbejde om projekter og tilbyder værktøjer til automatisering af bygge, test og deployment. Docker Hub er populært på grund af sin evne til at gøre udviklingen nemmere og mere effektiv.<sup>29</sup>

# Projektetablering

---

<sup>24</sup> Lucid shardt: <https://www.lucidchart.com/pages/product>

<sup>25</sup> GitHub: <https://docs.github.com/en/get-started/learning-about-github/githubs-products>

<sup>26</sup> Codespace: <https://docs.github.com/en/codespaces/overview>

<sup>27</sup> GitHub Actions: <https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>

<sup>28</sup> Docker: <https://www.docker.com/resources/what-container/>

<sup>29</sup> DockerHub: <https://docs.docker.com/docker-hub/>

## Prioritering af problemstillingerne

De 2 problemstillinger er vægtede ligeligt, der ud over er der lavet en prioritering inden for de 2 problemstillinger. Prioriteringerne bliver beskrevet i afsnit under.

## System til registrering af hvor artifact er deployed

1. Api
  - a. Bankdata har givet udtryk for at API delen af systemet er vigtigere end UI-delen, så derfor er den haft høje prioritet en denne.
2. UI

## Alternativ til Bankdata's CI/CD -pipelines

1. Github Codespace
  - a. Er blevet prioriteret over CI/CD da det vurderet at være den del som vil give flest nye muligheder, derud over er det også den del som vil kræve mest tid at sætte sig ind i.
2. CI/CD

I analysedellen har vi valgt ikke at lave en fuldstændig og komplet analyse, men kun valgt de dele af UML 2, som vi finder nødvendig for at forstå hvad systemet skal indeholde.

## Aftale med Bankdata

Det er aftalt med Bankdata at de støtter op om den Agile udvikling metode.

Dette gøres på følgende måde.

- Produkt owner
- 1-2 udvikler til, som altid kan kontaktes ved afklarende spørgsmål
- Møder kan aftales efter behov
- Vi kan være onsite de første 30 dage af projekt forløbet, hvilket er blevet brugt

Deployment System har integration til Bankdata Api'er, disse Api'er er interne og er derfor ikke udstillede på Internetet, det er derfor ikke muligt at test Deployment Systemet mod disse Api'er.

Bankdata leverer derfor et mockup af disse Api'er, som har et interface der matcher de rigtige Api'er. Bankdata laver den og de byggede med OpenApi go Swagger-UI, images leveres i DockerHub

- STF System tilhørs forhold
  - [mxchub/stf-service:<tag>](#)
- Info Change managed system
  - [mxchub/info-service:<tag>](#)
- Art Wrapper af Artifactory Api
  - [mxchub/info-service:<tag>](#)

# Valg af project form og arbejdsmetoder

## Agilt

Vi fandt en række fordele ved at anvende agile arbejdsprocesser. For det første øges fokus på værdiskabelse ved at levere værdi til kunderne så hurtigt som muligt gennem løbende leverancer af små, men færdige dele af et projekt.

For det andet opnås en bedre tilpasningsevne til ændringer, da agile processer muliggør, at organisationer kan reagere hurtigt på ændringer i forretningsbehov og markedet, da de er designet til at håndtere ændringer på en struktureret måde.

For det tredje øges medarbejdertilfredsheden, da agile metoder giver medarbejderne større indflydelse på deres arbejde og muliggør deres bidrag med ideer og løsninger på problemet, hvilket kan føre til øget medarbejdertilfredshed og motivation.

For det fjerde forbedres kommunikationen og samarbejdet, da agile processer fokuserer på løbende kommunikation og samarbejde mellem teammedlemmer, hvilket kan resultere i bedre samarbejde og kvaliteten af det endelige produkt.

Til sidst opnås en forbedret risikostyring, da agile metoder fokuserer på at identificere og håndtere risici i projektets tidlige faser, hvilket kan reducere risikoen for problemer senere i processen.<sup>30</sup>

## Scrum Kanban

Som en enkeltperson kan scrum og kanban anvendes til at styre og organisere projekter.

For at gøre dette, er der først og fremmest definere målene og prioriteringskriterierne for projektet. Derefter blev projektet opdeles i mindre dele, kaldet "user stories", og planlægges ind i sprints i en uge af gangen. Det var også være hensigtsmæssigt at oprette et kanban-bræt for at visualisere arbejdsflowet og holde styr på fremskridtet. På kanban-brættet blev opgaverne trukket gennem arbejdsflowet fra "Ready For refinement" til "Done", og der er implementeres limiter for at begrænse mængden af arbejde, der er i gang på et givet tidspunkt, dog har det ikke været til meget brug efter at teamet kun har bestået af en person, så typisk har der ikke være flere en et par opgaver igang på sammen tid. Samt pull-systemer for at styre, hvilke opgaver der skal tages op næste. Det er vigtigt at huske, at både scrum og kanban er fleksible metoder, der kan tilpasses til de specifikke behov og krav, man har for sit projekt.

## Sprint planning

Nogle årsager til at der er blevet brugt sprint planning er for at sikre, at teamet har en klar forståelse af de mål og prioriteringer, der skal opnås i løbet af sprinten. Samt at sikre, at teamet har en klar roadmap for, hvad der skal gøres i løbet af sprinten, og at de kan fokusere på at levere værdi for kunden så hurtigt som muligt. Kanban-bræt er blevet brugt til visualisering af sprintet <sup>31</sup>

## Roadmap

Roadmap / epic er blevet brugt til at beskrive de store mål og visioner for et projektet. Roadmappen har hjælpe med at give et overblik over, hvad der skal opnås i løbet af en

---

<sup>30</sup> agilt: <https://planaprojects.com/da/agile/>

<sup>31</sup> sprint planning: <https://www.scrum.org/resources/what-is-sprint-planning>

længere periode over 2 måneder, og har hjulpet med at sætte retningen for, hvilke opgaver og aktiviteter der skal prioriteres.<sup>32</sup>

### Dayli scrum meeting

Som en solo udvikler blev det fundet nyttigt at holde daglige stand-up møder, selvom at det er et solo team. Dette hjælp solo teamet med at holde fokus og holde styr på ens fremskridt, samt give en mulighed for at reflektere over, hvad der gik godt og hvad der kan forbedres. Dayli scrum blev brugt dagligt, i starten af projektet, blev det ikke brugt særlig meget, dog i produktions fasen gav det et godt ind blik hvor langt man var kommet og var man mangle eller ikke havde nået siden sidste møde.<sup>33</sup>

### Sprint review

Selvom teamet kun bestod af en enkelt person, fandt enkels person teamet stadig det nyttigt at holde sprint review for at reflektere over ens fremskridt og at lave feedback på ens egen arbejde. Dog er det svært at finde fejl eller mangler i ens egen nuværende del af projektet<sup>34</sup>

### Kanban-bræt

Kanban-bræt er blevet brugt som et visualiserings Tool til arbejdes flowet og styre opgaverne i projektet, her er der blevet oprettede colonder med forskellige stager i, som beskriver hvor i processen, en givende produkt er i. det blev brugt til at holde styr på hvor langt man er i processen af udviklingen. Her under kan man bruge pull systemet, som giver den givende udvikler mulighed for at hurtig flyttet en produkt fra udvikling til testning og videre til færdig. Det spare teamet en deltid efter som om han den individuelle person selv kan styre fremgangen af hans produkt.<sup>35</sup>

## Valg og opsætning af project managements tools

### Jira

#### Hvorfor Jira

Jira er et projektstyringsværktøj, der er udviklet til at hjælpe teams med at planlægge, spore og udføre opgaver i et projekt. Det er særligt anvendeligt i forbindelse med brugen af scrum og kanban-metoder, da værktøjet indeholder specifikke funktioner, der understøtter disse metoder. Dette skiller sig ud i forhold til et andet projektstyringsværktøj, Trello, hvor man selv skal opsætte hele funktionaliteten af sit kanban-board eller epics. De to nævnte funktioner var de must-have-funktioner, der skulle være til rådighed i det projektstyringssystem, der blev valgt, og derfor blev Jira valgt som det system, der skulle anvendes i teamet.

---

<sup>32</sup> raodmap / epic: <https://kanbanize.com/kanban-resources/portfolio-kanban/portfolio-flow-efficiency>

<sup>33</sup> dayli scrum meeting: <https://www.scrum.org/resources/what-is-a-daily-scrum>

<sup>34</sup> sprint rewiev: <https://www.scrum.org/resources/what-is-a-sprint-review>

<sup>35</sup> Kanban-bræt: <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-board>



Brugen af et værktøj til arbejdsgange, såsom Jira, kan bidrage til at forbedre teamets effektivitet ved at give et klart, organiseret rammeværk til at styre og spore arbejdet.

## **Hvordan har jeg brugt jira**

### **Konfiguration af jira**

Under projektetablering skulle der også oprettes et workflow til teamets kanban-borde. Jira har et fantastisk værktøj til at visualisere teamets workflow. Dette værktøj indeholder grå, blå og grønne bokse, der fungerer som kolonner i kanban-bordet.

### **Ready For refinement**

I denne fase opretter man sin historie i kanbanboard, under oprettelsen diskutere man om hvad historien omhandler, hvilke kriterier der er for udførelsen af historien, om der er mulige sub-stories der skal oprettelse til den.

Dette brugte jeg flittigt det første måned da jeg stadig sad i bankdata, her brugte jeg muligheden for at side og diskutere de forskellige stories med en anden udvikler, som arbejde på deres del af det samlet system.

### **In refinement**

I In refinement fasen udføre man de valg der er taget på grundlag af de diktionser der var i den tidligere fase. Efter ind udførelse af beslutningerne videre sende den To Do fasen

### **To Do**

I denne fase indgår opgaven i et projektstyringsværktøj som en "To Do"-opgave, indtil den er klar til at blive påbegyndt af en udvikler. Dette kan være, når der er tildelt en udvikler til opgaven, eller når der er opstået en passende mulighed for at påbegynde den.

### **In progress**

Når en opgave bliver tildelt en udvikler og påbegyndes, flyttes den til fasen "In progress". I denne fase forventes det, at udvikleren arbejder på opgaven, indtil den er færdiggjort eller flyttes til en anden fase.

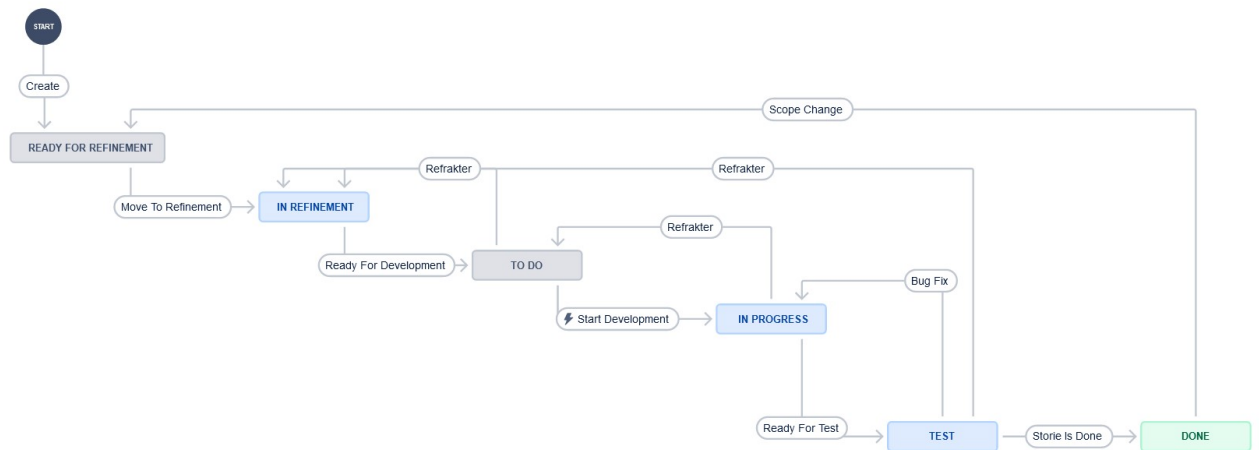
### **Test**

I fasen "Test" bliver produktet testet efter det er blevet færdigudviklet og flyttet ud af fasen "In progress". Der kan udføres mange forskellige typer af tests i denne fase, såsom enhedstest, integrationstest, godkendelsestest og regressionstest, for at sikre, at produktet fungerer som forventet og lever op til de specifikationer, der er blevet fastsat for det.

I vores tilfælde har vi brugt build test i form af Github Actions til at teste produktet. Dette hjælper os med at sikre, at produktet er fejlfrit og lever op til de krav, vi har sat for det, inden det sættes i drift. Det er vigtigt at teste produktet grundigt, så vi undgår fejl eller mangler

### **Done**

Formålet med fasen "Done" er at sikre, at opgaven eller featuren er færdigudviklet og klar til at blive implementeret i det endelige produkt, og at der ikke er nogen videre udvikling eller forbedringer, der er nødvendige. Dette hjulpet med at holde projektet på sporet og sikre, at der ikke opstår forsinkelser eller uforudsete udfordringer.

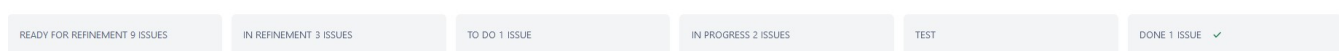


I billedet kan der ses en visuel illustration af workflowet. I illustrationen kan der ses 3 forskellige farve bokse.

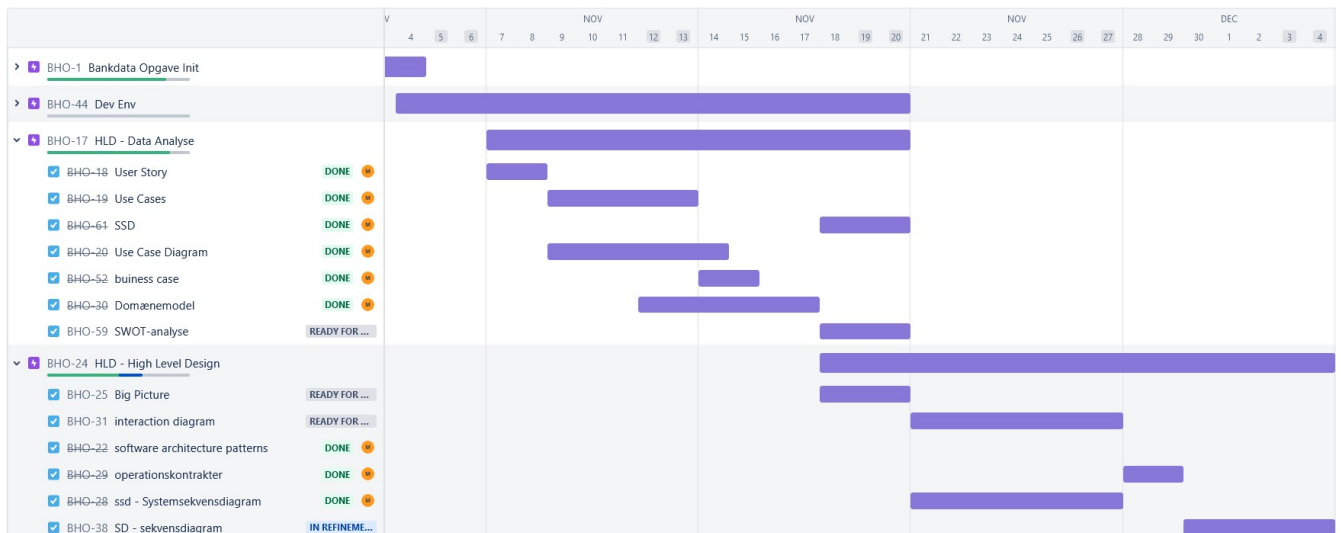
De grå bokse symboliserer opbevaringsbokse, hvor opgaverne holdes indtil de skal arbejdes på. Dette kan have været nyttigt, når der er nogen form for konceptuel udvikling, der skal udføres, før opgaven kan begynde at blive arbejdet på. Det har også været nyttigt, når der var en række forudsætninger, der skal være opfyldt, inden opgaven kan begynde at blive arbejdet på.

De blå bokse symboliserer udviklingsbokse, hvor opgaverne arbejdes på. Fasen "In refinement" indeholder opgaver, der er under opstilling, mens fasen "In progress" indeholder opgaver, der er under udvikling. Dette har været nyttigt at vide, da dette har givet et overblik over, hvilke opgaver der er under udvikling, og hvilke der er klar til at blive testet.

Den grønne og sidste boks er fasen "Done", der kun indeholder produkter, der er færdige og klar til at blive afleveret til kunden eller en anden interessent. Dette er den sidste fase i projektet, og det indebærer, at opgaven eller featuren er blevet færdigudviklet og testet, og er klar til at blive implementeret i det endelige produkt.



I denne illustration kan man se, hvordan opgaverne flyttes fra kolonne til kolonne i takt med, at de gennemgår forskellige faser i projektet. For eksempel vil en opgave begynde i kolonnen "ready for refinement", når den er blevet tilføjet til kanban-boardet, og så flyttes den til kolonnen "In Progress", når den begynder at blive arbejdet på. Når opgaven er færdigudviklet og testet, vil den blive flyttet til kolonnen "Done".



Jira har været en meget nyttig hjælp til at holde styr på projekttiden. Funktionen har gjort det muligt for mig at overholde den tidsramme, der er sat for mit projekt, samt for hver enkelt epic. Dette har været særligt nyttigt, da det har gjort det muligt for mig at sikre, at jeg ikke overskrider den tid, der er afsat til projektet eller epics, samt at holde styr på, om jeg holder mig inden for de fastsatte deadlines.

Jiras funktion til administration af opgaver på ethvert givent tidspunkt har også været en stor hjælp. Den har gjort det muligt for mig at altid have en plan for, hvad jeg skal arbejde på, samt holde styr på, hvilke opgaver der skal udføres, og hvilke der er afsluttet. Alt i alt har Jira været en meget nyttig ressource til at holde styr på projekttiden og opgaverne, og jeg har haft stor gavn af at bruge værktøjet.

## Confluence

Confluence er et samarbejds- og dokumentationsværktøj, der gør det muligt at oprette, redigere og organisere dokumenter og indhold på en centraliseret platform. Det er nyttigt til at holde styr på og dele information inden for en organisation eller et team

I dette projektarbejde har Confluence været et nyttigt værktøj til strukturering og organisation af noter. Især har page-funktionaliteten været nyttig, da den har muliggjort en struktureret organisering af noterne. Dette har gjort det nemmere at finde og få adgang til relevant information på et senere tidspunkt. Derudover har det været nyttigt at kunne linke noterne til de relevante epic eller stories i Confluence, da dette har gjort det nemmere at overskue, hvad der er relevant for hvert enkelt projekt. Desuden har muligheden for at dele noterne med andre udviklere samt uploade artefakter i Confluence været en fordel, da dette har gjort det nemmere at samarbejde med andre og dele vigtig information om projekterne. Alt i alt har Confluence været en meget nyttig hjælp i dette projektarbejde, da det har muliggjort en struktureret og effektiv organisering og deling af noter og andre dokumenter.

# Valg af udvikling sprog og tool change

## Udvikling sprog og frameworks

Programmering sprog og relaterede frameworks er valgt efter ønske fra Bankdata da de efterfølgende skal kunne overtage source og arbejde vider med systemet. Dette gælder for følgende.

- Java 17
- Frontend Node.js
- Microservice (API) Quarkus

## Github

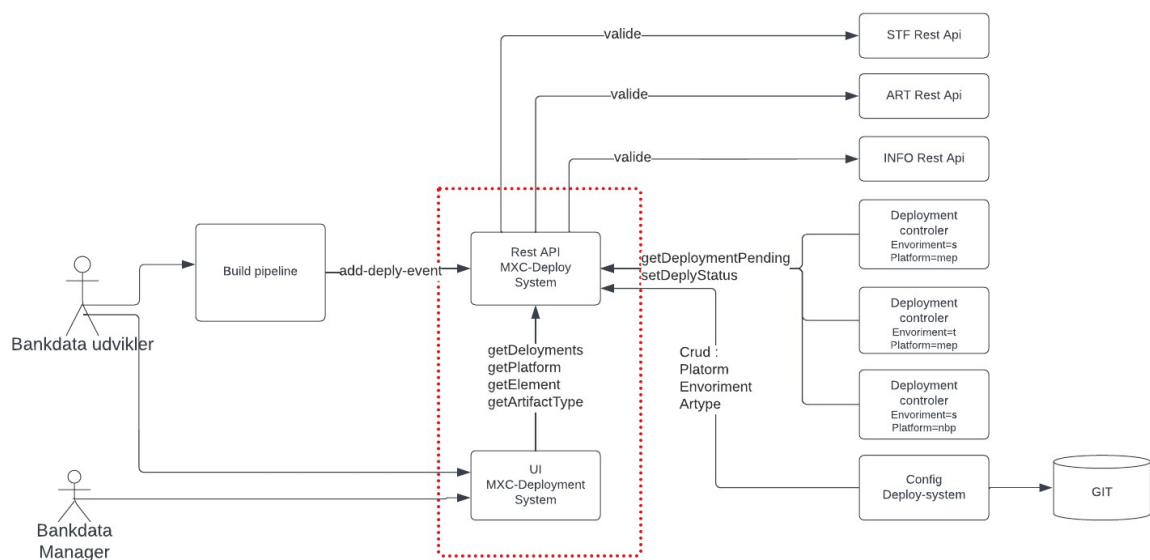
Er valgt til kode repository.

## Visual Studio Code

Er valgt som IDE at det har en meget tæt og god integration til GitHub Codespaces

# System

Der er lavet et system diagram for at vise i hvilken sammen hæng MXC Deployment Systemt skal ind gå i. MXC Deployment Systemet er indrammede af en rød stiplede linje. Alle interaktioner ind og ud af systemet skal implementeres som Rest Api kald. UI skal indeholde oplysninger vedr. Deployment, metadata som Platform, Environment og ArtifactType skal ikke være en del af UI, men det skal være muligt at udføre CRUD funktioner via Rest Api kald.



# Analyse

Et standard context diagram

## User stories

### Story nr. 1

**Som** deploy ansvarlig

**Vil jeg** kunne enkelt eller masse registrere hvad der skal deployes

**Så** jeg udføre deployments pga. disse registreringer, samt finde ud af hvor en artifact er blevet deployet

#### Mulige udfald:

Platform finde ikke

Miljø findes ikke

Artifact type findes ikke

Source host findes ikke (git, artifactory)

Url findes ikke

STF findes ikke, hvis det er udfyldt ikke findes

Navn og version findes allerede

### Story nr. 2

**Som** deploy ansvarlig / deploy controller

**Vil jeg** kunne finde frem til hvilke artefakter der er deployet på en platform, miljø og for en given artifact type

**Så** jeg kan gendeploy alle artifacts på platformen

#### Mulige udfald:

En liste af navn og current version (1.1.1 eller sha)

## Use cases

### Case nr. 1

**Mål:** At kunne nemt eller masseregistrere hvad der skal deployes, så deployments kan udføres på grundlag af disse registreringer, samt finde ud af, hvor en artifact er blevet deployet

<b>Title:</b>	<b>Registrer deployments</b>
<b>Aktør:</b>	CI pipeline

<b>Senerio</b>	<ol style="list-style-type: none"> <li>1. indsend de nødvendige oplysninger, inklusive platformen, miljøet, artifact-typen, URL'en, samt navn og version af artifacten</li> <li>2. Send registreringen</li> </ol>
----------------	---

**Forventet resultat:** Deploymentet registreres succesfuldt og kan udføres på grundlag af denne registrering. Systemet er i stand til at finde ud af, hvor artifacten er blevet deployet.

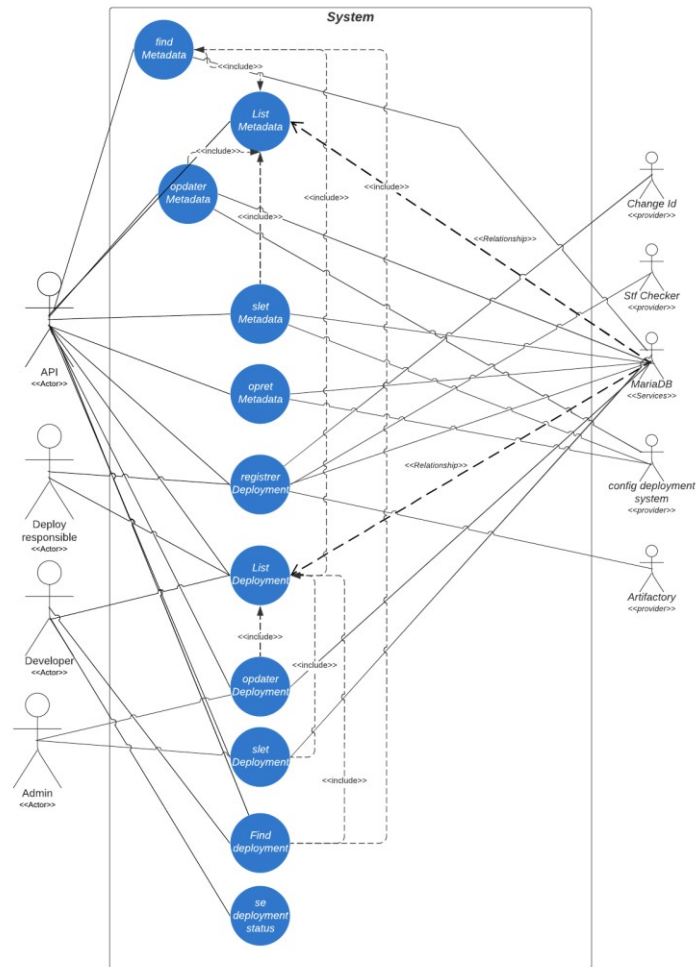
## Case nr. 2

**Mål:** At kunne finde frem til hvilke artifacts der er deployet på en given platform, miljø og for en given artifact-type,

<b>Title:</b>	<b>Find deployed artifacts</b>
<b>Aktør:</b>	udvikler
<b>Senerio</b>	<ol style="list-style-type: none"> <li>1. gå ind i deployment-registry-platformen</li> <li>2. Gå til siden for deployment</li> <li>3. Vælg platformen, miljøet, og artifact-typen</li> <li>4. Se listen over url'er, platformen, miljøet, og artifact-typen og version for deployed artifacts</li> </ol>

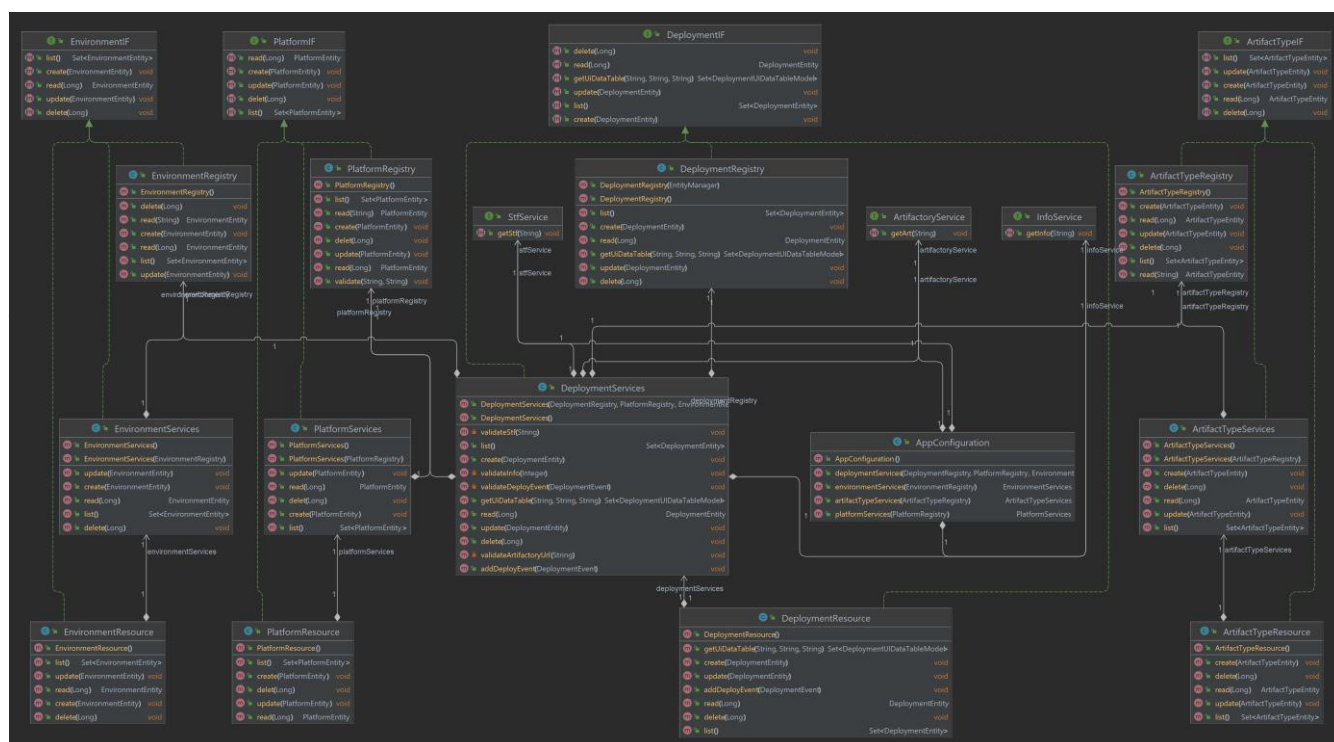
**Forventet resultat:** Der vises en liste over deployed artifacts,

Use Case Diagrams (Behavioral)



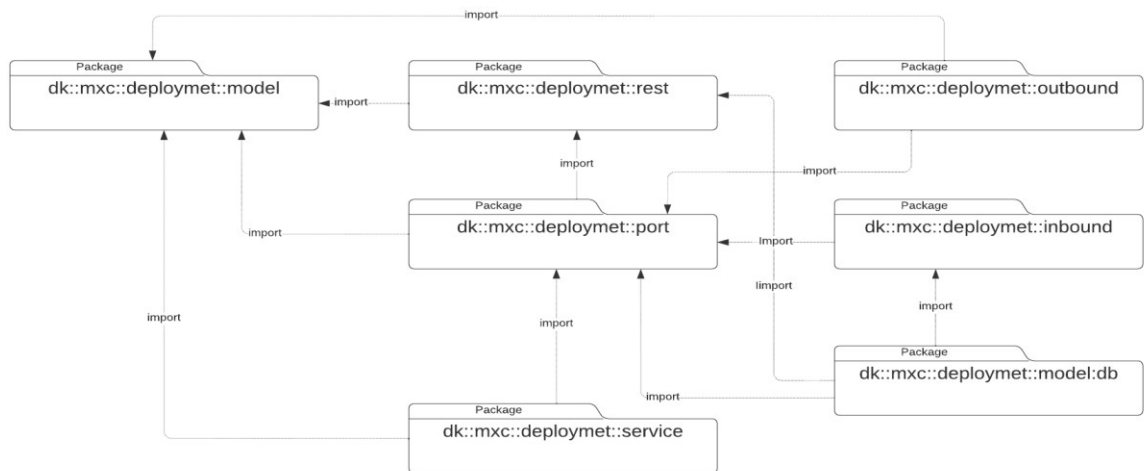
## Component Diagram (Structural)

## Class Diagrams (Structural)



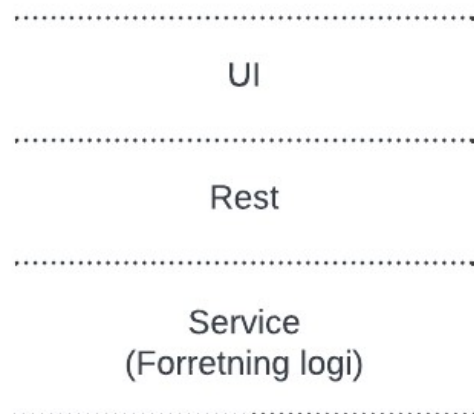
- Inboundt
  - Indholder classer der er lavet som registries, disse er en facade for databasen eller andre bakends.
- Outbound
  - Indholder rest clienter til eksterne endpoints
- Model
  - Indholder classer som er de data strukturer som bruges i systemet. Roden af model pakken indholder data modler som ikke er en del af data modellen
  - Sub pakke db indholder alle Entity datamodeller som bruges mod databasen.





## Lagdelt systemdesign

En lav delt arkitektur er en type softwarearkitektur, hvor komponenterne i systemet er designet til at være så uafhængige af hinanden som muligt. I systemer er denne lag delte arkitektur valgt for at opnå, nemmere udvikling, teste og vedligeholde af systemet, da hver komponent kan behandles uafhængigt af de andre.

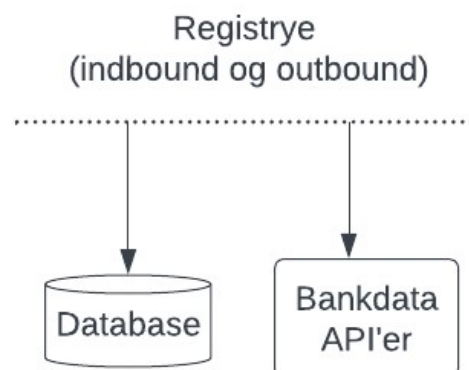


## Object Diagrams (Structural)

## Interaction Diagram (Behavioral)

Livscyklus for deployment

status = not deployed, deployed, undeployed, dropped



## Sequence Diagram (Behavioral)

## Implementering

I dette afsnit beskrives og gennemgås forskellige dele af koden for at beskrive hvordan de enkelt funktioner er implementeret. Koden der tager udgangspunkt i er klasser til backend api Deployment og så koden til UI.

### Hvordan få man adgang til coden og codespaces

Der er lavet gæste bruger til Github, så man kan logge ind og se coden, brugerne har også rettigheder til at bruge codespace.

Brugrid	Password	For hvem
mxc-dev-1	jZVY3HHLxJcBtZAUjBDp	Lære
mxc-dev-2	v1EHUV7jU9jy22sUfui4	Sensorer

Koden der gennemgås kan findes i følgende repositories.

- <https://github.com/mgc-org/hog-deployment-registry>
- [https://github.com/mgc-org/Angular\\_env](https://github.com/mgc-org/Angular_env)
- <https://github.com/mgc-org/hog-mariadb-image>

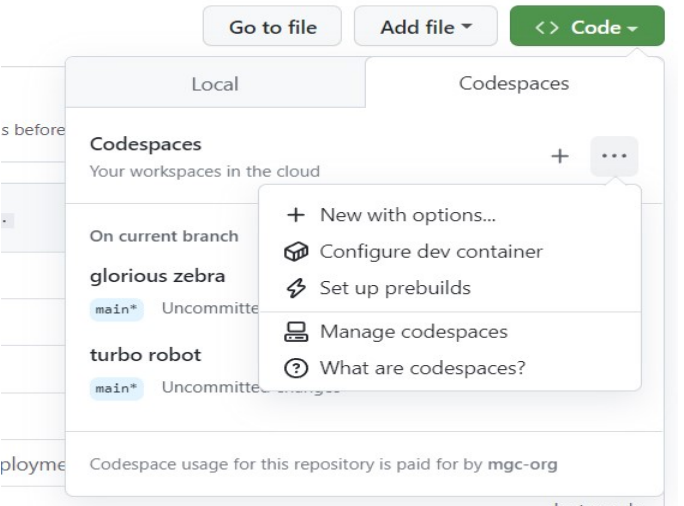
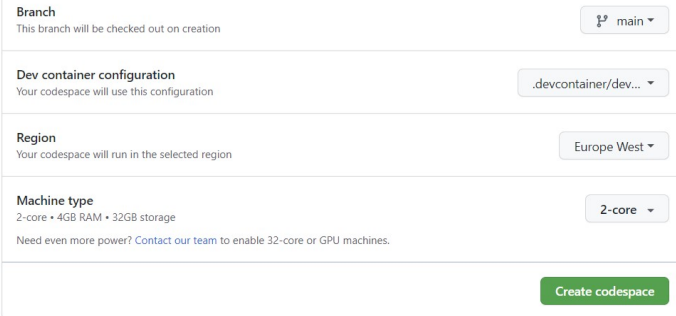
### Hvordan tester man det

Begge de beskrevne metoder til at få adgang til codespace, starter codespacet op i browseren. Hvis man vil bruge vscode skal man installere GitHub Codespaces extension.<sup>36</sup>

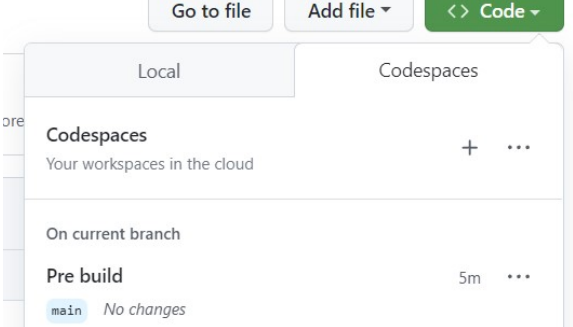
### Create new codespace

---

<sup>36</sup> Codespace in browser eller vscode: <https://docs.github.com/en/codespaces/developing-in-codespaces/using-github-codespaces-in-visual-studio-code>

<ol style="list-style-type: none"> <li>1. Tryk på grøn knap</li> <li>2. Vælg New with options...</li> </ol>	
<ol style="list-style-type: none"> <li>1. Ændre Machintype til 4 cores</li> <li>2. Tryk grøn knappe Create codespace</li> </ol>	

## Genbrug et pre build code space

<ol style="list-style-type: none"> <li>1. Tryk på grøn knap</li> <li>2. Under current branche vælg klik på teksten Pre build</li> </ol>	
---	--

## Hvordan start man programmet

For HOG-deployment-service

1. I venstre menu vælg Gradle (ikon elefan)
2. I Gradle menu tryk på Tasks
3. Åben quarkus submenu
4. Dobbelt klik quarkusDev

Hvis gradle ikke er korrekt initeret med menu.

- Tryk på tre prikker ud for Gradle Projects
- Tryk på Refresh Gradle Projects View
- Afvendt at menu refresher

I terminal kan man nu følge opstart af applikationen, når man ser dette output med teksten **Started in** er applikatione startede.

```

", "jar", "d", "Deployed", "2022-12-18 12:07:00")
Hibernate:
INSERT INTO deployment(id, url, version, changeId, stf, fk_platformShortName, fk_artifactType, fk_environmentShortName, status, statusUpdated) VALUES (32, "test.url", "1.0.1345", "12345678", "ose", "
nbp", "jar", "t", "Pending", "2022-12-18 20:24:00")
Hibernate:
INSERT INTO deployment(id, url, version, changeId, stf, fk_platformShortName, fk_artifactType, fk_environmentShortName, status, statusUpdated) VALUES (33, "test.url", "1.1.1", "12345678", "ose", "nbp
", "jar", "t", "Failed", "2022-12-18 21:59:00")
Hibernate:
INSERT INTO deployment(id, url, version, changeId, stf, fk_platformShortName, fk_artifactType, fk_environmentShortName, status, statusUpdated) VALUES (34, "test.url", "1.0.1", "12345678", "ose", "ose
", "ear", "t", "Deployed", "2022-12-18 12:07:00")
Hibernate:
INSERT INTO deployment(id, url, version, changeId, stf, fk_platformShortName, fk_artifactType, fk_environmentShortName, status, statusUpdated) VALUES (35, "test.url", "1.0.1345", "12345678", "ose", "
nbp", "jar", "t", "Pending", "2022-12-18 20:24:00")
Hibernate:
INSERT INTO deployment(id, url, version, changeId, stf, fk_platformShortName, fk_artifactType, fk_environmentShortName, status, statusUpdated) VALUES (36, "test.url", "1.1.1", "12345678", "ose", "ose
2022-12-26 08:22:45,184 INFO [io.quarkus] (Quarkus Main Thread) hog-deployment-registry 1.0.0-SNAPSHOT on 70h (powered by Quarkus 2.14.3.Final) started in 6.727s. Listening on: http://localhost:8080
2022-12-26 08:22:45,184 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2022-12-26 08:22:45,185 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [agroal, cdi, hibernate-orm, jdbc-mariadb, narayana-jta, rest-client, rest-client-jackson, resteasy, resteasy-jackso
n, smallrye-context-propagation, smallrye-metrics, smallrye-openapi, swagger-ui, vertx]
=====--> 90% EXECUTING [13m 42s]
> IDE
> :quarkusDev
> IDE
> IDE
  
```

Til højre for tabel terminal tryk på taben port

Port	Local Address	Running Process	Visibility	Origin
5005	https://ubisgc-potential-doodle-r7...	/usr/lib/jvm/java-17-openjdk-amd64/bin/java -Dquarkus.test.basic-consol...	Private	Auto Forwarded
8080	https://ubisgc-potential-doodle-r7...		Private	GitHub Codespaces
8081	https://ubisgc-potential-doodle-r7...	/usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8081 -contain...	Private	GitHub Codespaces

Før musen op på linien med port 8080, under LocalAdresse er det et browser ikon tryk på det og appen vil nu være til gængelig i browseren.

Pga fejl i codespace, ser det ikke altid ud som om at porten er aktiv.

QUARKUS

# Congratulations!

You just made a Quarkus application.

This page is served by Quarkus.

[VISIT THE DEV UI](#)

This page: `src/main/resources/META-INF/resources/index.html`  
App configuration: `src/main/resources/application.properties`  
Static assets: `src/main/resources/META-INF/resources/`  
Code: `src/main/java`  
Generated starter code

[RESTEasy JAX-RS](#) Easily start your RESTful Web Services  
→ [SPatch](#) `@Spatch`  
→ [Related guide](#)

Application

GroupId: io.quarkus  
ArtifactId: quarkus-quarkus-registry  
Version: 1.0.0-SNAPSHOT  
Quarkus Version: 1.14.2.Final

Selected extensions

• RESTEasy Classic Jackson

[Documentation](#)  
Practical step-by-step guides to help you achieve a specific goal. Use them to help get your work done.

[Set up your IDE](#)  
Everyone has a favorite IDE they like to use to code. Learn how to configure yours to maximize your Quarkus productivity.

Tryk nu på blå knap **Visit the dev UI**

Dev UI

hog-deployment-registry

Build

Build Steps

Configuration

Config Editor  
Dev Services

ArC

Build time CDI dependency injection  
Beans: 72  
Observers: 2  
Interceptors: 1  
Removed Components: 73

Datasources

Configure your datasources  
Reset Databases

Hibernate ORM

Define your persistent model with Hibernate ORM and JPA  
Persistence units: 1  
Entities: 5  
Named Queries: 0

SmallRye Metrics

Expose metrics for your services  
All Metrics  
Vendor Metrics  
Application Metrics  
Base Metrics

SmallRye OpenAPI

Document your REST APIs with OpenAPI - comes with Swagger UI  
OpenAPI  
Swagger UI

I boksen SmallRye OpenAPI tryk på link Swagger UI

Swagger UI

q/openapi

Explore

hog-deployment-registry (powered by Quarkus)

hog-deployment-registry API

4.0.0-SNAPSHOT

GA2D

Artifact Type Resource

POST /artifacttype/create

DELETE /artifacttype/delete/{id}

GET /artifacttype/list

GET /artifacttype/read/{id}

PATCH /artifacttype/update

Deployment Resource

GET /deployment/UIDataTable/{platformShortName}/{environmentShortName}/{artifactType}

POST /deployment/add-deployment

Hvis man har lagt at bruge vscode og GitHub Codespaces extension kan appen tilgås på <http://localhost:8080/q/swagger-ui/>

## Backend Rest API

### Rest Api

Denne klasse repræsenterer en ressource, der er tilgængelig på netværket gennem en webapplikation. Klassen er markeret med en `@Path`-annotation, der angiver, at denne klasse vil håndtere webforespørgsler ved hjælp af Quarkus og RESTEasy. Derudover er JSON-serielisering/deserielisering håndteret af RESTEasy Jackson. Klassen implementerer et interface kaldet `DeploymentAPI` og har et felt kaldet `deploymentServices`, som indeholder en reference til en instans af

DeploymentServices-klassen, der er indsat ved hjælp af dependency injection i Quarkus. Derudover er der en metode kaldet `create`, der håndterer HTTP POST-forespørgsler og kalder metoden `create` på en instans af DeploymentServices-klassen med et DeploymentEntity-objekt som parameter.

For at denne klasse vi virke skal følegnde implemnteringer med tages i gradl.build filen.<sup>37</sup>

implementation 'io.quarkus:quarkus-resteasy-jackson'

implementation 'io.quarkus:quarkus-resteasy'

```
22
23 @Path("/deployment")
24 public class DeploymentResource implements DeploymentIF {
25
26     @Inject
27     DeploymentServices deploymentServices;
28
29     @Override
30     @POST
31     @Path("/create")
32     public void create(DeploymentEntity deploymentEntity) {
33         deploymentServices.create(deploymentEntity);
34     }
35
36 }
```

Rest Client kald

Denne kode definerer et interface kaldet **InfoService**, der repræsenterer en RESTful webtjeneste.

**@Path**-annotationen angiver, at denne tjeneste vil være tilgængelig på stien `"/info"`.

**@RegisterRestClient**-annotationen anvendes til at angive, at dette interface skal registreres som en "REST klient" ved hjælp af Quarkus' REST klient-funktionalitet.

Dette betyder, at Quarkus vil oprette en instans af tjenesten ved runtime og gøre den tilgængelig for andre dele af programmet.

Der er også en metode i interfacet kaldet **getInfo**, der er markeret med **@GET**-annotationen, hvilket angiver, at denne metode vil håndtere HTTP GET-forespørgsler. Derudover er metoden markeret med **@Path("/get/{changeNo}")**, hvilket angiver, at denne metode vil håndtere forespørgsler, der matcher stien `"/info/get/{changeNo}"`.

**@Produces**-annotationen angiver, at denne metode vil producere et svar i form af JSON.

**@PathParam**-annotationen anvendes til at angive, at der er en path parameter kaldet `"changeNo"` i stien, og at denne parameter skal indsættes i metoden som en parameter kaldet `"changeNo"` af typen **String**.

Alt i alt vil denne kode definere et interface, der repræsenterer en RESTful webtjeneste, der er tilgængelig på stien `"/info"` og håndterer HTTP GET-

---

37 Rest API: <https://quarkus.io/guides/resteasy>

forespørgsler til stien "/info/get/{changeNo}", producerer et svar i form af JSON og tager en path parameter kaldet "changeNo" som input. Tjenesten vil blive registreret som en REST klient ved hjælp af Quarkus' REST klient-funktionalitet.<sup>38</sup>

```

1 package dk.mxc.deployment.outbound;
2
3 import javax.ws.rs.GET;
4 import javax.ws.rs.Path;
5 import javax.ws.rs.PathParam;
6 import javax.ws.rs.Produces;
7 import javax.ws.rs.core.MediaType;
8
9 import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;
10
11 @Path("/info")
12 @RegisterRestClient
13 public interface InfoService {
14     @GET
15     @Path("/get/{changeNo}")
16     @Produces(MediaType.APPLICATION_JSON)
17     void getInfo(@PathParam("changeNo") String changeNo);
18 }

```

## JPA – Hibernate

39

```

@Entity
@Table(name="deployment")

//@NamedQuery(name = "findById", query = "SELECT id, uri, status, statusUpdated, fk_platformShortName, fk_artifactType, fk_environmentShortName FROM deployment WHERE id = :id")
public class DeploymentEntity implements Serializable {

    @Id
    @GeneratedValue(generator = "deploymentSeq", sequenceName = "deployment_id_seq", allocationSize = 1, initialValue = 100)
    @GeneratedValue(generator = "deploymentSeq")
    private Long id;
    private String uri;
    private String version;
    private Integer changeId;
    private String stff;
    private String status;
    @JsonIgnore
    private Timestamp statusUpdated;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "fk_platformShortName", referencedColumnName = "platformShortName")
    private PlatformEntity platform;
}

```

## Interface

40

---

38 Rest Client kald: <https://quarkus.io/guides/rest-client>

39 JPA – Hibernate: <https://quarkus.io/guides/hibernate-orm>

40 Interface: [https://www.w3schools.com/java/java\\_interface.asp](https://www.w3schools.com/java/java_interface.asp)



```

1 package dk.mxc.deployment.port;
2
3
4 import java.util.Set;
5
6 import dk.mxc.deployment.model.DeploymentUIDataTableModel;
7 import dk.mxc.deployment.model.db.DeploymentEntity;
8
9 public interface DeploymentIF {
10
11     public void create(DeploymentEntity deploymentEntity);
12     public DeploymentEntity read(Long id);
13     public void update(DeploymentEntity deploymentEntity);
14     public void delete(Long id);
15     public Set<DeploymentEntity> list();
16     public Set<DeploymentUIDataTableModel> getUIDataTable(String platformShortName, String environmentShortName, String artifactType);
17 }
18
19
22
23 @Path("/deployment")
24 public class DeploymentResource implements DeploymentIF {
25
26     @Inject
27     DeploymentServices deploymentServices;
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

## Metriker

For opfyldes kravet identificer i følgende user stories, er der lavet en løsning som leverer Prometheus metrick fra alle endpoints i systemet.

- Se statistikker om systembrug (se i bilag)
- Se systemoplysninger (se i bilag)

## Applikation

På applikations siden er det løst ved at bruge en Quarkus extension `quarkus-smallrye-metrics`<sup>41</sup>. Ved at bruge to anoteringer kommer der metrick med antal request og response tider.

```

72 @Override
73 @GET
74 @Path("/list")
75 @Counted(name = "ListCount", description = "How many list have been performed.")
76 @Timed(name = "ListTimer", description = "A measure of how long it takes to perform the list.", unit = MetricUnits.MILLISECONDS)
77 public Set<DeploymentEntity> list() {
78     return deploymentServices.list();
79 }
80

```

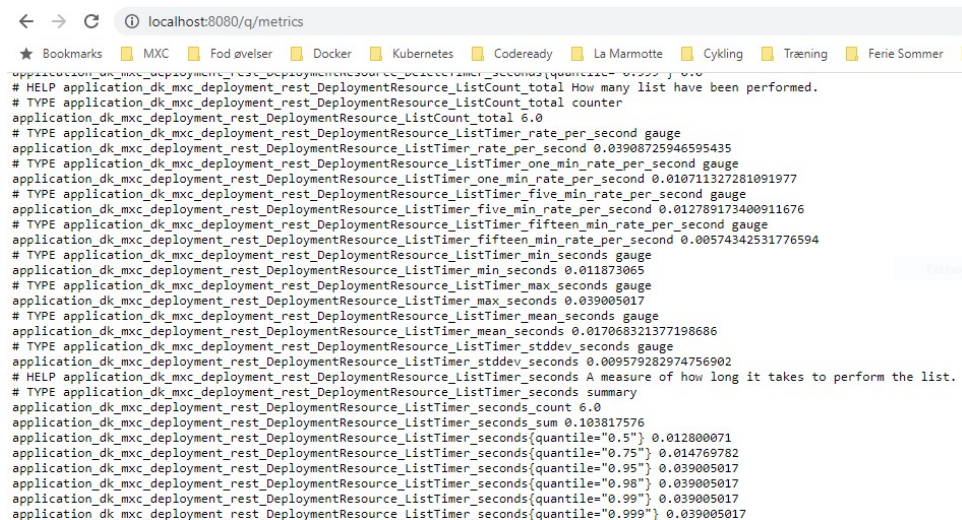
<sup>41</sup>Smallrye Metrics: <https://quarkus.io/guides/smallrye-metrics>



## dk.mxc.deployment.rest.DeploymentResource

Brugen af denne extension gør også at der kommer metricer om Java JVM heap, threads, cpu. Metricerne kan ses ved at til gå uri

- “/q/metrics” alle metricer fra applikationeen
- “/q/metrics/vendor” JVM oplysninger
- “/q/metrics/application” metricer lavet pr endpoint via kode eksemple
- “/q/metrics/base” JVM og maks skin oplysninger



```
← → ↻ localhost:8080/q/metrics
★ Bookmarks MXC Fod øvelser Docker Kubernetes Codeready La Marmotte Cykling Træning Ferie Sommer
# HELP application_dk_mxc_deployment_rest_DeploymentResource_ListCount_total How many list have been performed.
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListCount_total counter
application_dk_mxc_deployment_rest_DeploymentResource_ListCount_total 6.0
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_rate_per_second gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_rate_per_second 0.03908725946595435
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_one_min_rate_per_second gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_one_min_rate_per_second 0.010711327281091977
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_five_min_rate_per_second gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_five_min_rate_per_second 0.012788173400911676
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_fifteen_min_rate_per_second gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_fifteen_min_rate_per_second 0.00574342531776594
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_min_seconds gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_min_seconds 0.011873065
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_max_seconds gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_max_seconds 0.039005017
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_mean_seconds gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_mean_seconds 0.017068321377198686
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_stddev_seconds gauge
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_stddev_seconds 0.009579282974756902
# HELP application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds A measure of how long it takes to perform the list.
# TYPE application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds summary
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds_count 6.0
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds_sum 0.103817576
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds(quantile="0.5") 0.012800071
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds(quantile="0.75") 0.014769782
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds(quantile="0.95") 0.039005017
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds(quantile="0.98") 0.039005017
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds(quantile="0.99") 0.039005017
application_dk_mxc_deployment_rest_DeploymentResource_ListTimer_seconds(quantile="0.999") 0.039005017
```

## Kubernetes

For at metricerne kan komme ind i Pormetheus og dermed bruges i Grafana til at laver de ønskede grafer, skal Prometheus konfigurationen opdateret til at scrape end-pointer udestillede af applikationen. Dette gøres ved at tilføje en ServiceMonitor til Helm chartet som installer applikatione på Kubernetes. ServiceMonitor er en custom resource som installeres af Pormetheus-operateren. ServiceMonitore skal applyes til det namespace som Prometheus-operatoren er installeret for at den kan se resourcen, derefter skal der være en namesspaceSelector som matcher namespace hvor Services findes i dette tilfælde “mxs” og så skal der være selector der matcher en label app.kubernetes.io/instance: “hog-service”<sup>42</sup>

<sup>42</sup> Using Service Monitors: <https://observability.thomasriley.co.uk/prometheus/configuring-prometheus/using-service-monitors/>

```

1  apiVersion: monitoring.coreos.com/v1
2  kind: ServiceMonitor
3  metadata:
4    labels:
5      serviceMonitorSelector: prometheus
6    name: hog-service
7    namespace: monitor
8  spec:
9    endpoints:
10   - interval: 30s
11     targetPort: 8080
12     path: /metrics
13   namespaceSelector:
14     matchNames:
15     - mxc
16   selector:
17     matchLabels:
18       app.kubernetes.io/instance: "hog-service"

```

Applikatione vil nu være konfigureret som et target i Prometheus.

serviceMonitor/monitor/hog-service/0 (1/1 up) [Show logs](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<a href="http://10.244.0.3:8080/q/metrics">http://10.244.0.3:8080/q/metrics</a>	<span>UP</span>	<a href="#">container="hog-service"</a> <a href="#">endpoint="8080"</a> <a href="#">instance="10.244.0.3:8080"</a> <a href="#">job="hog-service"</a> <a href="#">namespace="mxc"</a> <a href="#">prometheus="prometheus-kube-prometheus"</a> <a href="#">prometheus="prometheus-kube-prometheus"</a>	2.303s ago	49.107ms	

Frontend UI

## GitHub Codespace

### Bankdatas situation i dag

Gennem praktikophold på Bankdata og samlaler med Bankdata ansatte er der fundet frem til følgende problem stillinger ved udvikler maskiner på Bankdata.

### FAK konceptet

Udvikler maskiner på Bankdata leveres fra JN-data<sup>43</sup>, konceptet kaldes FAK. FAK koncepter er et koncept hvor man kan bygge maskiner og opdatere software på disse. Der findes forskellige software pakker man kan vælge i mellem.

<sup>43</sup> JN-data: <https://www.jndata.dk/>

- Kunde rådgiver i bankerne
- Software udviklere i Bankdata
- Leder, Produkt owner, Konsulenter i Bankdata

Fældes for alle profiler er dog at bygges og administreres under det samme FAK koncept. For en Bankdata udvikler er der en række problemer med FAK konceptet.

## Låst maskine

Maskinen er låst, hvilket betyder at man ikke kan installere software eller konfigurationer. Dette kan dog omgås ved via Sikkerheds afdeling at få retighed til en funktion mednavn Acting Admin. Ved at aktivere denne funktion kan man så installere software. Når man aktiverer acting admin er den aktiv i 30 minutter, hvor efter den skal forlænges, hvis dette ikke sker og man er midt i en install eller en fejl søgning, kan det have fatale konsekvenser og man må starte forfra.

## Inrollede på Bankdata network

Hvilket giver diverse problemer.

- Maskinen kan kun nå internettet gennem proxy og Bankdata's proxy er påtvunget en meget restriktiv profil, hvilket har følgende konsekvenser.
  - Kan ikke installere software
  - IDE'er kan ikke hente plugin og extensions
  - Kan ikke bruge offentlige repositories så f.eks *mvnrepository.com*, *dette kan dog omgås ved at bruge Bankdata's onprim Artifactory*.
- *Kan ikke bruges på offennetwork som har en anden VPN end Bankdata's*

## Genetablering af maskiner

Hvis udvikler kommer galt afsted og ikke kan få deres set up til at virke, er der som regel kun en udvej og det er at reinstallere maskinen. Det er en nem proces for det er det FAK konceptet er bygget til. Problemet er bare at det man får en maskine tilbage med den default profil der var valgt. Det er nu op til udvikleren at reetablere alle de software pakker og konfigurationer der skal til for at de kan lave deres arbejde og det kan være en meget tung proces.

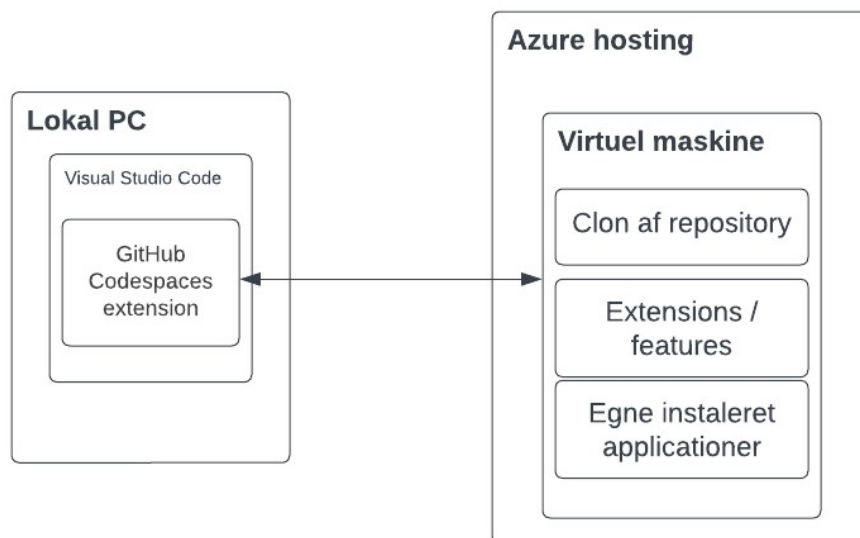
## Hvad er GitHub codespace

"GitHub Codespaces er et skybaseret udviklingsmiljø, der gør det muligt for udviklere at skrive, køre og fejlfinde kode direkte i webbrowseren. Det inkluderer integreret understøttelse af Git, en kodeeditor og en terminal, så udviklere kan arbejde på deres kode uden behov for lokale værktøjer eller opsætning.

I projektet er det valgt at bruge Visual Studio Code som har en stærk integration til GitHub Codespace, hvor udvikleren kan oprette, åbne og administrere deres Codespaces. Udvidelsen tilbyder også integrerede værktøjer til at oprette, klon, commit og push ændringer til et GitHub-repositorium, så udvikleren kan samarbejde med andre og administrere deres kode på en struktureret måde.

Generelt set gør udvidelsen det nemmere for udviklere at arbejde med GitHub Codespaces direkte fra Visual Studio Code, uden at skulle skifte mellem værktøjer eller bruge en separat webbrowser. Det giver også mulighed for at få adgang til alle de andre produktivitetsværktøjer, der følger med Visual Studio Code, såsom syntaksfremhævning, debugging, testning og mere.

Det er vigtigt at bemærke, at selv om man bruger Visual Studio Code til at få adgang til Codespace, er det stadig en separat cloudbaseret service, der kører på en virtuel maskine i skyen.<sup>44</sup>



Når man opretter en GitHub Codespace, har man mulighed for at vælge en skabelon, der passer til behovet. Skabelonerne dækker forskellige programmeringssprog og teknologier, så man kan vælge den, der passer bedst til projektet. Når man har valgt en skabelon, vil den indeholde et sæt af forud installerede værktøjer og biblioteker, der er relevante for det valgte programmeringssprog og de teknologier, der er inkluderet i skabelonen. Dette gør det nemmere at komme i gang med at skrive kode uden at skulle installere og konfigurere alt selv.

Det er også muligt at tilpasse Codespace ved at installere ekstra værktøjer og biblioteker, der ikke er inkluderet i skabelonen. Dette kan gøres ved hjælp af en terminal eller en anden form for kommandolinjeværktøj. Man kan også tilpasse indstillingerne i Codespace ved at ændre konfigurationsfiler eller bruge andre værktøjer til at tilpasse arbejdssted.

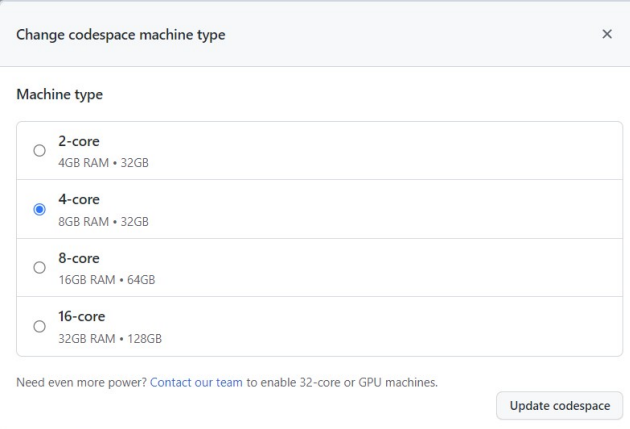
På denne måde kan man påvirke, hvordan din Codespace er sat op og konfigureret, så den passer bedst til dine behov og arbejdsgange. Man kan efter at have konfigureret et codespace gøre det til en template som så kan bruges af andre med samme behov.<sup>45</sup>

<sup>44</sup> Developing inside a Container: <https://code.visualstudio.com/docs/devcontainers/containers>

<sup>45</sup> GitHub Codespaces: <https://docs.github.com/en/codespaces>

## Dimensionering af codespace

Når man opretter et codespace kan man vælge mellem forskellige størrelser af maskiner, man kan efterfølgende ændre denne størrelse, hvis det viser sig at man har valgt for lidt eller for meget.



Change codespace machine type

Machine type

- ☐ 2-core  
4GB RAM • 32GB
- ☒ 4-core  
8GB RAM • 32GB
- ☐ 8-core  
16GB RAM • 64GB
- ☐ 16-core  
32GB RAM • 128GB

Need even more power? [Contact our team](#) to enable 32-core or GPU machines.

Update codespace

## Custom configuration af codespace

Det er muligt at konfigurere et codespace lige som man ønsker det både med extensions til ens IDE og med features som specifikke Java versioner, Angular osv.

Configuration af codespaces kan gøres ved først at de standard features og extensions, hvorefter man kan bede om at få dem gemt i `.devcontainer`, dette gør at der i roden af repositoryet laves en folder med navn `.devcontainer`.<sup>46</sup> I denne folder er der en Docker file og `devcontainer.json`, disse filer indholder konfigurationen og er en del af det flow der køres under rebuild og connect. I `devcontainer.json` er der mulighed for at hugges sig ind i forskellige stadier af flowet omkring build og connect, hvilket er en meget brugbart når man ønsker at lave specifikke codespace der passer lige til de behov man har i udvikling processen <sup>47</sup>

## Rebuild

Hvis man ændre dimensionering på maskinen eller man ændre i konfigurationen under `.devcontainer` folder, er der behov for et rebuild. Rebuild er en operation hvor codespacet bygges og gemmes som et docker image, det er så dette images der bruges når man fremover connecter til et codespace, hvilket gør opstart tiden på et codespace meget hurtigere. Alle filer i folderen `/workspaces`, som er der man normalt arbejder når man er i et codespace, de påsigeres bevares under et rebuild og et full rebuild<sup>48</sup>

---

<sup>46</sup> Creating a custom dev container configuration: <https://docs.github.com/en/codespaces/setting-up-your-project-for-codespaces/introduction-to-dev-containers#creating-a-custom-dev-container-configuration>

<sup>47</sup> [https://containers.dev/implementors/json\\_reference/#\\_devcontainerjson-properties](https://containers.dev/implementors/json_reference/#_devcontainerjson-properties)

<sup>48</sup> Rebuild: <https://docs.github.com/en/codespaces/codespaces-reference/performing-a-full-rebuild-of-a-container>

## Port forwarding

Når codespace er en virtuel maskine der køre i skyen, hvordan kan man så teste f.eks en rest service som er udviklede og startede i codespace på <http://localhost:8080>.

Codespace har en port forward funktion som automatisk opdager at der lyttes på en port og laver en forward til den lokale maskine som er connectede til codespacet. Det er også muligt at pre konfigurere denne port forwaring i devcontainer.json filen.<sup>49</sup>

## Design og implementering af codespaces

I projektet er der blevet identificeret 2 udvikler profiler Frontend og Backend , de to udviklere profiler bruger vidt forskellige tools, så det er oplagt at lave en codespace opsætning til hver af dem.

I projekt vil vi også gerne skabe en integration / verifikations miljø som et codespace, dette codespace skal bruge til at verificere at alle krav til systemet er blevet løst. Det er her man kan verificere de krav der har være til løsning som ikke har være funktionelle krav, her tænkes på følgende ting stade i problemformuleringen.

- Kubernetes som afvikling platform
- ArgoCD som CD
- Grafana og Prometheus til metricer

Efter at det er blevet identificeret hvilke udvikling profiler der skal være i projekte, har man set på er hvilke værktøjer de enkelte udvikler profil skal bruge for at kunne løse deres opgaver.

---

<sup>49</sup> Forwarding ports in your codespace: <https://docs.github.com/en/codespaces/developing-in-codespaces/forwarding-ports-in-your-codespace>

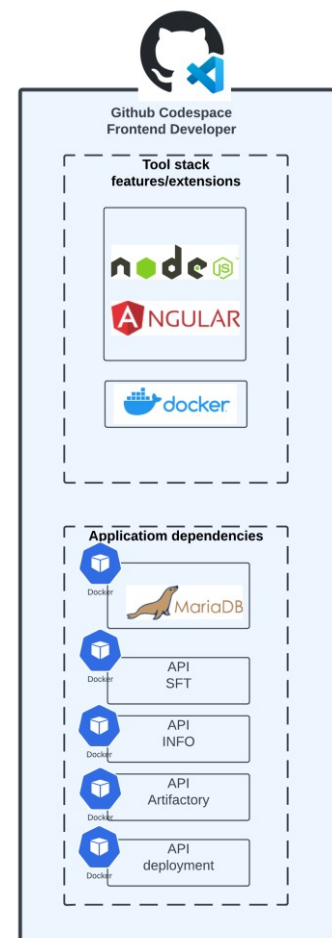
## Codespace til Frontend udviklere

Frontend udvikleren har brug for følgende too og applikationer for at kunne udvikle UI til systemet.

- Node.js
- Angular
- Docker – docker til at køre backends og database

Frontend udviklerene skal også kunne kalde de backends som UI bruge for at kunne test. Dette er opnået ved at starte følgende applikationer op som docker applikationer.

- MXC Deployment Systemet
- Mockup af Bankdata Api da de bruges af MXC Deploymentsystemet
  - STF-API
  - Info-API
  - Artifactory-API
- MariaDB med databasen Deployment\_db



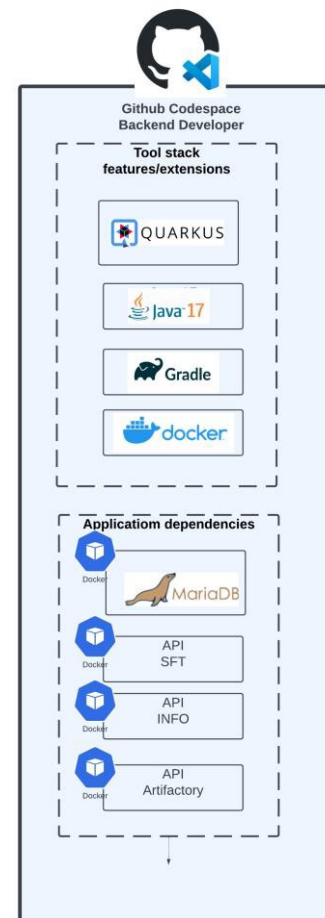
## Codespace til Backend udviklere

Backend udvikleren har brug for følgende tool og applikationer for at kunne udvikle API'er til systemet.

- Quarkus – CLI
- Java 17
- Gradle – CI build
- Docker

For at kunne teste backend systemet skal der være adgang til.

- MariaDB – med database Deployment\_bd
- Bankdata Api da de bruges af MXC Deploymentsystemet
  - STF-API
  - Info-API
  - Artifactory-API





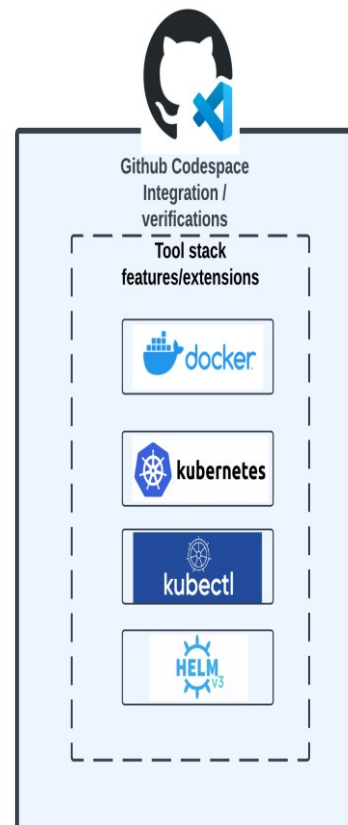
## Codespace til integrations / verifikations miljø

Integraions og vefikations miljøet, har en primær kompunet Kubernetes, den valgt kubernetes implementaion er Kind. Der udover er der en række værktøjer til at hjælpe med at mange en Kubernetes installation.

- Docker
- Kind – Kubernetes implementation
- Kubectl
- Helm

I Kind vil der bleive installer en række applikationer, disse applikationer installeres som Helm charts. Den første applikation der installeres er Argocd som er CD delen af CI/CD, efter den er installer så er det denne applikation der stå for at installere følgende applikationer.

- Grafana
- Prometheus
- MariaDB – vores eget image med Deployment\_db
- MXC-Deployment-Systeme
  - Backend - Api
  - Front end – UI
- Mockup af Bankdata Api da de bruges af MXC Deploymentsystemet
  - STF-API
  - Info-API
  - Artifactory-API



## Implentering af codespaces

Beskrivelsen af implementering tager udgang punkt i codespacet til Backend udvikler. Alle codespace er opbyggede efter den samme skabelon som er den der beskrives i codespace dokumentation.

For at kunne konfigurere egne codespaces skal der findes en .devcontainer folder i roden af repositoryet. I .devcontainer findes det som default to filer Dockerfile og devcontainer.json.<sup>50 51</sup>

Den styrende fil er

devcontainer.json har en

reference til Docker filen. Docker filen er en docker build file og når man rebuilder codespace er den den som bygger det docker image som er codespacet. Man definerer i Docker filen hvilke base image man vil bygge på de codespace der bruger i dette project bygger på er en



<sup>50</sup> Implementing af codespaces:

[https://containers.dev/implementors/json\\_reference/#\\_devcontainerjson-properties](https://containers.dev/implementors/json_reference/#_devcontainerjson-properties)

<sup>51</sup> Implementing af codespaces: <https://code.visualstudio.com/remote/advancedcontainers/overview>

Ubuntu 20.04 LTS (Long-Term Support), som er bygget til at kunne skabe connection mellem vscode og den virtuelle developer maskine.

```
3 # [Choice] Ubuntu version (use ubuntu-22.04 or ubuntu-18.04 on local arm64/Apple Silicon): ubuntu-22.04, ubuntu-20.04, ubuntu-18.04
4 ARG VARIANT="jammy"
5 FROM mcr.microsoft.com/vscode/devcontainers/base:0-${VARIANT}
6
```

Man kan bruge alle docker build commandor i denne file, her bruger vi den til at installer den version af Java vi skal bruge i projectet.

```
7 # [Optional] Uncomment this section to install additional OS packages.
8 RUN apt-get update && export DEBIAN_FRONTEND=noninteractive \
9     && apt-get -y install --no-install-recommends openjdk-17-jdk
10
```

I devcontainer.json findes der en række extension points som vi pre og post commands kan på virke codespace. Her er følgende brugt

- Features til at installer
  - Gradle
  - docker-in-docker
- customizations:vscode:extensions til at installer vscode extensions
  - vscjava.vscode-gradle
  - vscjava.vscode-java-pack
  - Redhat.java
- PostStartCommand til at køre script efter container er started
  - bash .devcontainer/install-services.sh

```
19
20 "postStartCommand": "bash .devcontainer/install-services.sh",
21
22
23 // Comment out to connect as root instead. More info: https://aka.ms/vscode-remote/containers/non-root.
24 "remoteUser": "vscode",
25 "features": {
26     "gradle": "latest",
27     "ghcr.io/devcontainers/features/docker-in-docker:2": {}
28 },
29 "customizations": {
30     "vscode": {
31         "extensions": [
32             "vscjava.vscode-gradle",
33             "vscjava.vscode-java-pack",
34             "redhat.java"
35         ]
36     }
37 }
```

### Installe-services start script

Dette script er enskrift udviklede til at starte de services som skal bruge for at kunne teste den applikation der skal udvikles i codespace. Dette sker ved at start images som docker containere. Scriptet starter MariaDB og de tre Bankdata Mockups. For hver container testes det om den allerede er defineret, om den er defineret afhænger af om det er en første connect efter et rebuild eller om man connecter til et prebuildet codespace. Derefter testes der for om containeren kører, hvis ikke så startes den.

```

13 #
14 # Styring af mariadb-deployment
15 #
16 docker port mariadb-deployment | grep tcp
17 if [ $? != 0 ]; then
18     echo "mariadb-deployment not started let's start it"
19     docker run --detach --name mariadb-deployment -p 3307:3306 --env MARIADB_ROOT_PASSWORD=mx-root-pw -v /var/mariadb-deployment/data:/var/lib/mysql sgchub/mariadb:1.0.1647d0a
20 fi
21 docker ps | grep mariadb-deployment
22 if [ $? != 0 ]; then
23     echo mariadb-deployment " not started let's start it"
24     echo docker start mariadb-deployment
25     docker start mariadb-deployment
26     echo
27 fi

```

# CI / CD

## GitHub Actions (CI)

### Hvad er GitHub Actions

GitHub Actions er et værktøj, der gør det muligt for udviklere at automatisere deres softwareudviklingsworkflows. Med GitHub Actions kan du oprette workflows, der udføres som reaktion på bestemte hændelser i dit GitHub-projekt, såsom at pushe en kodeændring eller at et nyt problem opstår. Dette gør det nemmere for udviklere at bygge, teste og deploye deres kode, og gør det muligt for dem at skabe mere komplekse og avancerede workflows til deres projekter.

### Design og implementering af Actions

Resultate af at køre disse build pipelines skal være et docker image som er pushed til DockerHub. For at nå frem til dette resultat skal der laves to pipe lines en til Backend som er kodet i Java og en til Frontend som er en Angular App. Her beskrive de enkelt steps som findes i de to pipelines.

Actions er en del af source repository og de placeres i en folder i roden af repositoryet med navn .github/workflows i workflow folderen ligger der en fil som konfigurerer den pipeline der skal køre samt hvilke events som skal trigger at de køres.

Alle steps i build pipelines er laver ved at genbruge allerede udviklede actions, som ligger tilgængelige til brug for alle, der er ikke fundet behov for at udvikle egne actions til vores build piplines.

### Java Backend build pipeline (pipeLine.yml)

- Definer at der skal bygges på følgende evnets
  - create pull reques, dette gør at man kan nå at lave en test inde der pushes til master
  - Push til master
- Hvilke OS skal der bruges når den virtuelle buld container starte
  - ubuntu-latest
- Definer de steps der skal være i piplinen
  - Setup java-version
  - Build with Gradle
  - Set up Docker Build
  - Build and push

- Build køre på `./src/main/docker/Dockerfile.jvm` som er docker build feilen fra vores repository.

Angular Frontend build pipeline

## ArgoCd (CD)

### Hvad er ArgoCd

Argo CD er et erklærende, GitOps-baseret værktøj til kontinuerlig levering til Kubernetes. Det hjælper teams med at automatisere den måde, de administrerer deres applikationer og deres erklærende konfiguration, så live-miljøet altid er i overensstemmelse med den ønskede stat, der er erklæret i versionskontrol. Argo CD bygges på det åbne kildeprojekt Argo og er en del af Cloud Native Computing Foundation (CNCF)-økosystemet.

### Design og implementering af ArgoCd

# Konklusion

## Codespace

Hvad har vi lært

Hvordan kan det bruges i fremtiden

## Systemudvikling

## Project management

## Samarbejde med Bankdata

# Kildehenvisninger

## Links

- Bankdata hjemmeside
  - o [www.bankdata.dk](http://www.bankdata.dk)
- User stories
  - o <https://www.agilealliance.org/glossary/user-stories>
- use cases:
  - o <https://www.usability.gov/how-to-and-tools/methods/use-cases.html>
- use case diagram (S. 3 –10)
  - o <http://csis.pace.edu/~marchese/CS389/L9/Use%20Case%20Diagrams.pdf>
- class diagram:
  - o <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
- database design: Database concepts-Pearson (S. 261)
- Scrum:
  - o <https://www.scrum.org/resources/what-is-scrum>
- kaban:
  - o <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban>
- sprint review:
  - o <https://www.scrum.org/resources/what-is-a-sprint-review>
- Kanban-bræt:
  - o <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-board>
- Agilt:
  - o <https://www.atlassian.com/agile>
- Kind
  - o <https://kind.sigs.k8s.io/>
- Open JDK 17
  - o <https://openjdk.org/projects/jdk/17/>
- Angular
  - o <https://angular.io/guide/what-is-angular>
- JPA – Hibernate ORM
  - o <https://hibernate.org/orm/>
- Quarkus
  - o <https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus>
- OpenAPI
  - o <https://spec.openapis.org/oas/v3.1.0>
- Visual Studio Code
  - o [https://en.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)
- Jira
  - o <https://www.productplan.com/glossary/jira/>
- Confluence
  - o <https://www.atlassian.com/software/confluence/guides/get-started/confluence-overview#hosting-options>
- Lucid chart
  - o <https://www.lucidchart.com/pages/product>
- GitHub
  - o <https://docs.github.com/en/get-started/learning-about-github/githubs-products>
- Codespace
  - o <https://docs.github.com/en/codespaces/overview>


- GitHub Actions
  - <https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>
- Docker
  - <https://www.docker.com/resources/what-container/>
- DockerHub
  - <https://docs.docker.com/docker-hub/>
- agilt:
  - <https://planaprojects.com/da/agile/>
- sprint planning:
  - <https://www.scrum.org/resources/what-is-sprint-planning>
- raodmap / epic:
  - <https://kanbanize.com/kanban-resources/portfolio-kanban/portfolio-flow-efficiency>
- dayli scrum meeting:
  - <https://www.scrum.org/resources/what-is-a-daily-scrum>
- sprint rewiev:
  - <https://www.scrum.org/resources/what-is-a-sprint-review>
- Kanban-bræt:
  - <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-board>
- Codespace in browser eller vscode
  - <https://docs.github.com/en/codespaces/developing-in-codespaces/using-github-codespaces-in-visual-studio-code>
- Rest API
  - <https://quarkus.io/guides/resteasy>
- Rest Clienr kald
  - <https://quarkus.io/guides/rest-client>
- JPA – Hibernate
  - <https://quarkus.io/guides/hibernate-orm>
- Interface
  - [https://www.w3schools.com/java/java\\_interface.asp](https://www.w3schools.com/java/java_interface.asp)
- Smallrye Metrics
  - <https://quarkus.io/guides/smallrye-metrics>
- Using Service Monitors
  - <https://observability.thomasriley.co.uk/prometheus/configuring-prometheus/using-service-monitors/>
- JN-data
  - <https://www.jndata.dk/>
- Developing inside a Container
  - <https://code.visualstudio.com/docs/devcontainers/containers>
- GitHub Codespaces
  - <https://docs.github.com/en/codespaces>
- Creating a custom dev container configuration :
  - <https://docs.github.com/en/codespaces/setting-up-your-project-for-codespaces/introduction-to-dev-containers#creating-a-custom-dev-container-configuration>
  - [https://containers.dev/implementors/json\\_reference/#\\_devcontainerjson-properties](https://containers.dev/implementors/json_reference/#_devcontainerjson-properties)
- Rebuild
  - <https://docs.github.com/en/codespaces/codespaces-reference/performing-a-full-rebuild-of-a-container>
- Forwarding ports in your codespace :
  - <https://docs.github.com/en/codespaces/developing-in-codespaces/forwarding-ports-in-your-codespace>
- Implemennting af codespaces
  - [https://containers.dev/implementors/json\\_reference/#\\_devcontainerjson-properties](https://containers.dev/implementors/json_reference/#_devcontainerjson-properties)
  - <https://code.visualstudio.com/remote/advancedcontainers/overview>

## Bøger

- APPLYING UML AND PATTERNS - CRAIG LARMAN
- Database concepts-Pearson - Kroenke, David M\_Auer, David J

## Bilag

### Test bruger



### Opret din Google-konto

Fornavn  
Mikkel Grimshave

Efternavn  
Christensen

Brugernavn  
mxc.dev.1

@gmail.com


Du kan bruge bogstaver, tal og punktummer

Ledige: [mikkelgrimshave](#) [mikkelgrimshavec](#)  
[cmikkelgrimshave](#)

[Brug min nuværende mailadresse i stedet](#)

Adgangskode  
MxcDev01#

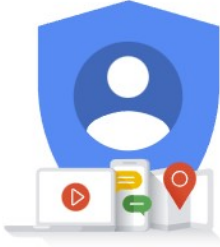
Bekræft  
MxcDev01#

 Angiv en stærkere adgangskode. Prøv med en kombination af bogstaver, tal og symboler.

☒ Vis adgangskode

[Log ind i stedet](#)

Næste



En konto. Dit helt eget Google.





## Opret din Google-konto

Fornavn  Efternavn

Brugernavn

Du kan bruge bogstaver, tal og punktummer

[Brug min nuværende mailadresse i stedet](#)

Adgangskode  Bekræft

Brug minimum 8 tegn med en blanding af bogstaver, tal og symboler

☒ Vis adgangskode

[Log ind i stedet](#)

Næste



Én konto. Dit helt eget Google.



## Velkommen til Google

mxc.dev.1@gmail.com



Telefonnummer (valgfrit)

Google bruger kun dette nummer til at beskytte din konto. Dit nummer kan ikke ses af andre. Du kan vælge senere, om nummeret skal bruges til andre formål.

Mailadresse til gendannelse (valgfrit)

Vi bruger den til at beskytte din konto

Dag  Måned  År

Din fødselsdag

Køn

[Derfor beder vi om disse oplysninger](#)

[Tilbage](#)

Næste



Vi beskytter dine personlige oplysninger

spørgsmål til møde

- tæller billeder med som text
- kan man kilde henvise til sit praktikophold
  - Bilag med doc i
- Spørg om problemformulering skal være præcis den samme som den der er godkendt
  - Det er den godkendte det skal med i
- Ssd omkring services actor
  - Det passer
- Minium anslag i rapport
- 
- 

<https://mgc-org.atlassian.net/jira/software/projects/BHO/boards/4/roadmap>

<https://mgc-org.atlassian.net/wiki/spaces/HOG/pages/950277/Opl+g+Datamatiker+hovedopgave>