

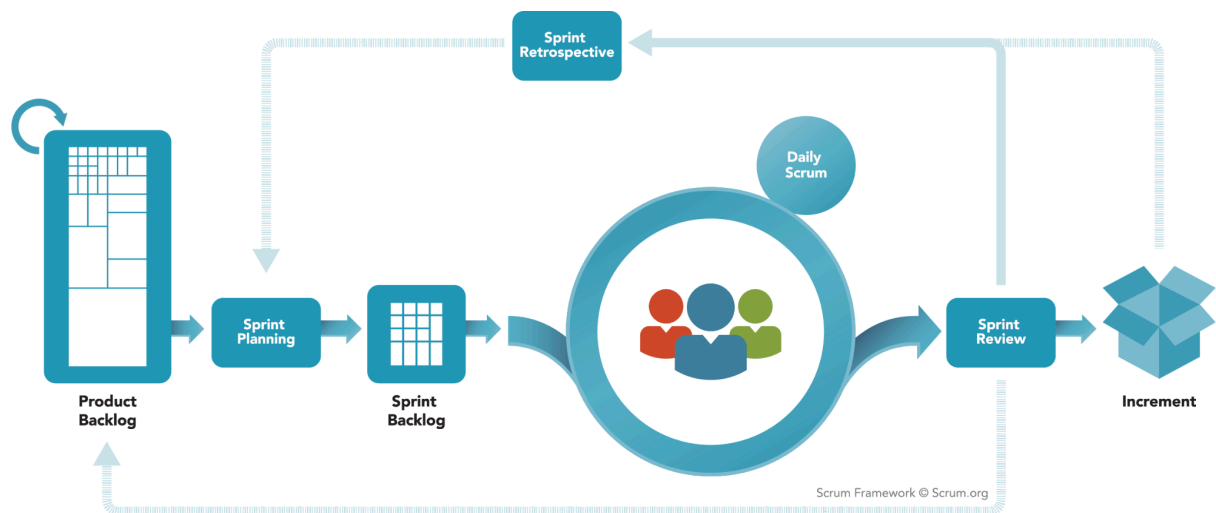
Minibasen site - Rapport

Gruppenavn: Gruppe 3

Navn på studerende: Andreas Østergaard Christoffersen, Christian Sverri Kruse & Mark Thuesen Nielsen

Skolens navn: Mercantec

Hold: Datatekniker med speciale i programmering - H1



Indholdsfortegnelse

Indledning	2
Problemstilling	2
Problemformulering	2
Tilgangsvinkel	2
Scrum	2
Prioriteret product backlog	3
Første sprint	3
Sprint planning	3
Sprint Backlog	4
Mockup	4
Database-diagram	7
Klassediagram	8
Kode	8
Sprint review	10
Sprint retrospective	11
Andet sprint	12
Sprint planning	13
Sprint Backlog	13
Sprint review	13
Sprint retrospective	13
Tredje sprint	14
Sprint planning	14
Sprint Backlog	14
Klassediagram	15
Sprint review	15
Sprint retrospective	15
Bilag	16
Bilag 1 - Gammel database-diagram - 1. sprint	16

Indledning

Vores product owner/kunde Peter, får udarbejdet en hjemmeside kaldet Minibasen. Peter bruger blandt andet hans hjemmeside som en markedsplads, hvor folk som har brug for at sælge eller købe, kan interagere med andre varer på siden. Peter kan bruge sin hjemmeside til at tjene penge på premium fordele eller reklamer. Det er denne opgave, som vi vil prøve at udføre i dette projekt.

Problemstilling

Vi har til opgave at udvikle salgsannonce funktioner til hjemmesiden Minibasen, heraf bl.a. salgsannonce oprettelse. I denne funktion skal brugeren kunne oprette et salgsannonce med et Bilnavn, Model, Årgang, Pris, Km kørt, Farve, Vægt, Rækkevidde, Drivmiddel, Gear type og Årlig skat skal derefter gemmes i en database, hvorfra brugeren senere kan hente sin salgsannonce igen og redigere i det.

Problemformulering

Vi vil arbejde med følgende spørgsmål i vores projekt.

- Brugeren skal have mulighed for at se alle de listede Cooperer.
- Brugeren skal kunne få en detaljeret oplysningsside i forhold til en bil.
- Brugeren skal kunne registrere/logge ind.
- Brugeren skal have CRUD muligheder i forhold til deres Cooperer.
- Man skal kunne relatere en Cooper til en bruger. Skal inkludere en "relativ" lokation i forhold til hvor man kan få bilen.
- (Måske) Bruger skal kunne "købe" en Cooper.

Tilgangsvinkel

Vi har haft et møde med kunden/product owner, hvorefter vi i gruppen har udarbejdet en problemstilling og problemformulering.

Vi vil herefter udarbejde en prioriteret product backlog, som vil beskrive de funktioner vi mener er essentielle, for at skabe de funktioner til kunde/product owners hjemmeside. Senere vil vi udarbejde nogle user stories, som vil beskrive de mest komplicerede funktioner.

Scrum

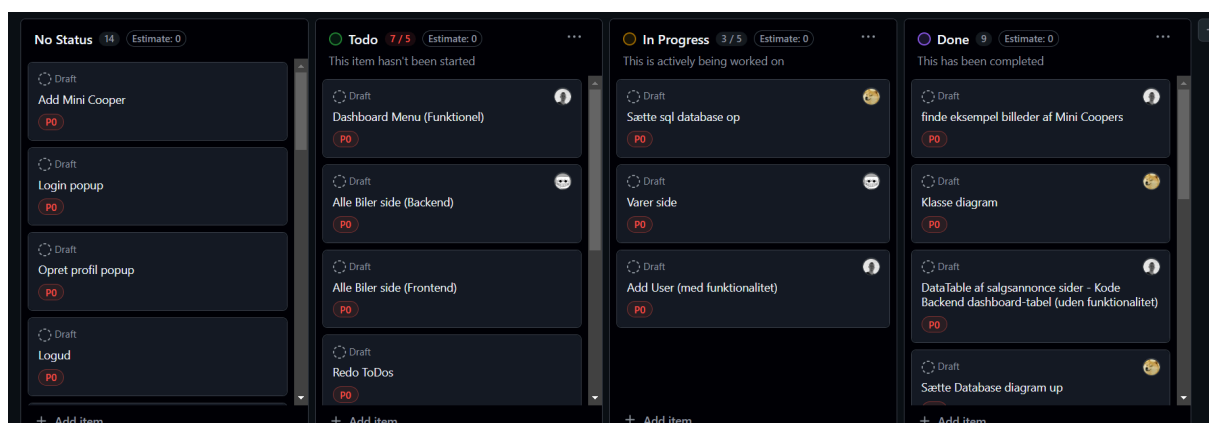
Som nævnt, skal vi i dette projekt bruge scrum som vores udviklingsmetode. Udviklingsmetoden er som sagt inkrementel og iterativ, hvilket betyder at vi arbejder i sprints, der denne gang har en varighed på en uge.

Scrum har roller som scrum master, product owner og developers - som har hver deres formål.

Rolle	Person(er)
Scrum Master	Andreas
Team/Developers	Christian Mark Thuesen Nielsen
Product owner Kunde	Peter

Prioriteret product backlog

Nu når vi har udarbejdet vores user stories, skal vi have lavet en prioriteret product backlog. Det er product owneren, som er ansvarlig for at få lavet vores prioriteret product backlog.



Første sprint

Første Sprint var fra d. 2/12 - 6/12 2024.

Sprint planning

For at kortlægge det arbejde vi vil lave i dette sprint, holder vi et sprint planning møde. Her finder vi i fællesskab hvilke user stories, som vi kan få at arbejde med i dette sprint. Vi har derefter udarbejdet et sprint goal - altså hvad målet er med dette sprint, og hvilken værdi ønsker vi at tilføje. Derefter opretter vi tasks til vores Prioriteret product backlog på "Github Projects", som er de opgaver der senere skal udføres, for at udføre vores user story og lave den til et inkrement i vores projekt.

Normalt vil et sprint planning møde varer otte timer, til et sprint forløb på en måned. Eftersom at vores sprints er kortere, nemlig en uge, vil vi afsætte to timere til vores sprint planning møde.

Sprint Backlog

Sprint backloggen vil indeholde de elementer, som vores team forventer at lave i løbet af sprintet og hvad vi hver især har valgt at arbejde med. Dette er i form af tasks, som er nødvendig at udføre for at levere funktionalitet til hjemmesiden. Disse task skal gennemføres, før det kan ændres fra en "To Do" task til et "done" inkrement.

I scrum backloggen er det muligt at tilføje nye tasks i løbet af et sprint. Dette sker i takt med at development teamet bliver klogere/får en bedre forståelse for, hvad der er nødvendigt at lave for at opnå teamets sprint goal. På samme måde kan development teamet fjerne elementer fra sprint backloggen, hvis elementet ikke længere er nødvendigt for at opnå sprint goalet. Disse ændringer kan kun laves af development teamet, i og med at det kun er dem som kan ændre i scrum backloggen i løbet af et sprint.

Grunden til at et sprint backlog er nyttig, er fordi det giver et løbende synligt billede af, hvad development teamets planer/arbejde er gennem sprintet.

Vi har oprettet et "Github Projects" til at skabe et overblik over vores projekt, som viser, hvad vi regner med at kunne nå i dette sprint. I dette "Github Projects" har vi tilføjet vores udarbejdede tasks som indeholder de funktioner, som vi skal programmere/implementere, samt modeller og diagrammer, brugertest, testning af kode og mockups.

Vi har valgt at tilføje følgende tasks, til vores første sprint i "Github Projects".

- Oprettelse af database
- Kode alle backend funktioner til "oprettelse af salgsannonce" siden
- Kode alle frontend funktioner til "oprettelse af salgsannonce siden
- database-diagram
- Klassediagram
- Low fidelity mockup til oprettelse af salgsannonce
- Brugertest til oprettelse af salgsannonce

Mockup

For at arbejde med den samme vision for produktet, har vi udarbejdet en mockup. Denne mockup er baseret på vores tilsvarende task til vores user story "Oprettelse af salgsannonce" i dette sprint, der kan visualisere hvorledes vores salgsannonce værktøj kan fungere/se ud. Således har teamet en fælles forståelse for, hvad vi arbejder imod. På samme tid med det kan vi kommunikere vores tanker hvis der er elementer som vi fælles ønsker at ændre.

Vælg kategori fra til

Vælg Model fra til

Biler til salg

Nyheder



[Hjem](#) [Produkter](#) [Om os](#) [Kontakt os](#)

[Login](#)

Login

Mail

Det er nødvendigt at udfylde en e-mail.

Kodeord

Det er nødvendigt at udfylde en kode.

Glemt din adgangskode?

Har du ikke profil? [Tilmeld](#)

Opret profil

Mail

Det er nødvendigt at udfylde en e-mail.

Kodeord

Det er nødvendigt at udfylde en kode.

Har du allerede en profil? [Login](#)

Opret salgsannonce

Bilnavn

Det er nødvendigt at udfylde bilnavn.

Model

Det er nødvendigt at udfylde model

Årgang

Det er nødvendigt at udfylde årgang

Pris

Det er nødvendigt at udfylde pris

Km kørt

Det er nødvendigt at udfylde km kørt

Farve

Det er nødvendigt at udfylde farve

Vægt

Det er nødvendigt at udfylde vægt

Rækkevidde

Det er nødvendigt at udfylde rækkevidde

Drivmiddel

Det er nødvendigt at udfylde drivmiddel

Gear type

Det er nødvendigt at udfylde gear type





Årlig skat

Det er nødvendigt at udfylde årlig skat

Vælg billede

Upload

Billeder



Sæt bil til salg

Opdater salgsannonce

Bilnavn

Det er nødvendigt at udfylde bilnavn.

Model

Det er nødvendigt at udfylde model

Årgang

Det er nødvendigt at udfylde årgang

Pris

Det er nødvendigt at udfylde pris

Km kørt

Det er nødvendigt at udfylde km kørt

Farve

Det er nødvendigt at udfylde farve

Vægt

Det er nødvendigt at udfylde vægt

Rækkevidde

Det er nødvendigt at udfylde rækkevidde

Drivmiddel

Det er nødvendigt at udfylde drivmiddel

Gear type

Det er nødvendigt at udfylde gear type


Årlig skat


Det er nødvendigt at udfylde årlig skat


Vælg billede


Upload

Billeder









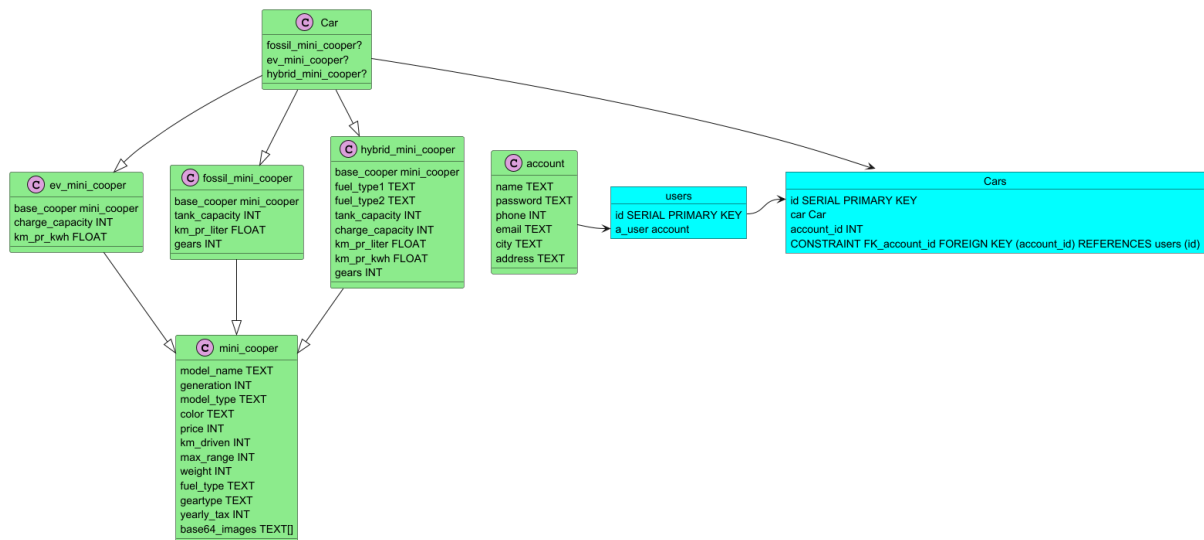
Opdater bil til salg

Database-diagram

Vi har nu lavet et database-diagram med den datastruktur, som vi er blevet enige om. Det er meningen, at database-diagrammet skal opfylde alle krav til hvilke data som database skal indeholde. Vi har aftalt i gruppen at den kan ændres senere, hvis vi i gruppen har fælles forståelse for, hvordan vi vil have databasen skal være konstrueret anderledes.

Dette er vores anden version af vores database-diagram, og vi har derfor tilføjet det gamle database-diagram til vores bilag¹. Vi har lavet det nye database-diagram, fordi vi har fået en bedre forståelse for hvordan man ændre database-diagrammet, så det ser mere overskueligt ud.

¹ Bilag 1 - Gammel database-diagram



Klassediagram

Vi har ikke udarbejdet et klassediagram til dette sprint. Vi har lavet noteret hvad der skal være i vores klassediagram, for at skabe en bedre forståelse for projektet, samt en enighed omkring, hvad vi skal tilføje for at opfylde vores user story.

BaseMiniCooper:

- BaseMiniCooper er den basale klasse for vores repræsentation af Mini Coopere.

Ev- Fossil- HybridMiniCooper:

- Ev- Fossil- HybridMiniCooper klasserne, er de forskellige type af Mini Coopere, i forhold til brændstoftype.

FullMiniCooper:

- FullMiniCooper er egentlig bare en klasse der implementerer de tre Mini Cooper klasser, hvor der kun kan være EN Mini Cooper, mens de andre to er null.

Kode

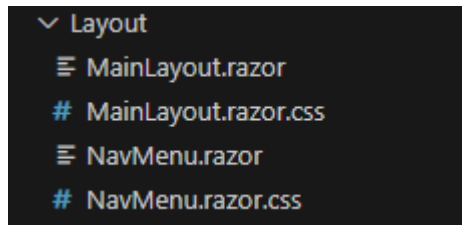
Vi har valgt at bygge selve platformen/web applikationen med Blazor applikation. Her udvikler vi med MVC (Model-View-Controller) design mønstret. Kort sagt, betyder dette, at vi separerer applikationen ind i tre grupper - modeller, views og controllers. Brugerens forespørgsler sendes til en controller, som bruger modellerne til at udføre disse, hvorefter et view vises til brugeren.

Den data der gemmes er oprettede salgsannoncer, brugerdata og så videre. Disse data bliver gemt i en SQL database. Denne database har vi oprettet, hvor vi derefter bruger Neon.Tech til at danne og udarbejde databasen med dens tilhørende tabeller osv. Derefter implementerer vi det i filen "DbContext.cs" til brug i applikationen.

Vores forbindelse til databasen er gemt i "appsettings.json" filen, som bruges når vi konfigurerer vores services i "Program.cs" filen.

Opbygning / Layout

Siden, visuelt set, er opbygget på "Components", og "Layouts"



Her ligger vores hjemmesides opbygning i Layouts.

Der er nogle hoved CSS og Razor elementer, som holder et "Tema" over vores side.

NavMenu, er vores "Header"



Her navigerer man mellem siderne og kommer tilbage til startsiden når der bliver trykket på MiniBasen Logo, eller "Home".

Sider er opbygget i lag, hvor MainLayout er opsætning, og NavMenu er header. Disse komponenter holder siden sammen, uanset hvor du er på hjemmesiden.

Mainlayout har altid en "<Body />", så den er altid det første der loader, hvor Navmenu følger.

Navmenu er bare en normal bootstrap Navbar, som er lavet om til en dynamisk header.

Logo og design var lavet af Mark i Figma, og så replikeret i Bootstrap af mig.

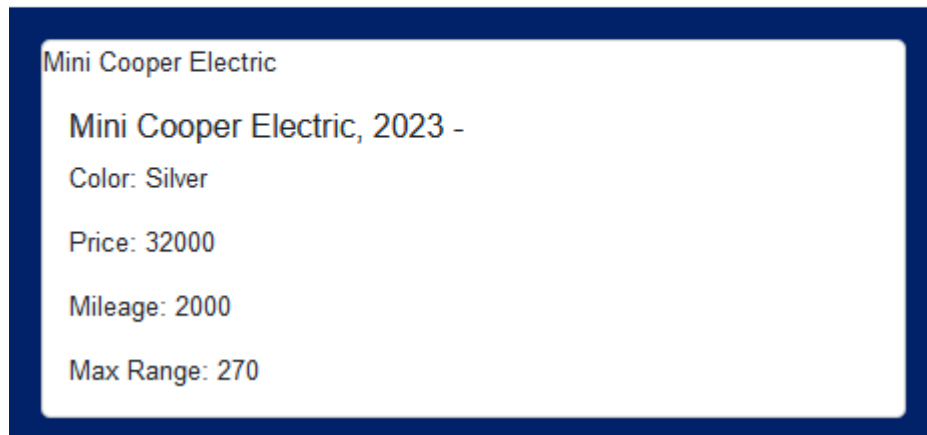
Siderne

Siderne er komponenter, som bliver lagt ind i MainLayout igennem "@page /", og det navn du nu vil bruge.

De har hver sit formål, og deler ikke ting, som Layout gør. Derfor, er meget af koden hver for sig, men samtidig flexibel.

"Cars" er bare en SQL loading mekanisme, der opbygger en liste over alle biler med cards, i bootstrap.

Mini Coopers



Her bruger det data fra vores SQL, med at bruge `@inject DBService DbService` og loade det ind med et "For each" som her:

```
@if (fullMiniCoopers == null)
{
    <p>Loading...</p>
}
else
{
    @foreach (var miniCooper in fullMiniCoopers)
```

Det lader os også bruge en lille "Loading" text, imens den loader listen af mini coopers fra vores SQL DB.

```
var baseCooper = miniCooper.GetEvCooper() as MiniCooper.BaseMiniCooper
??
miniCooper.GetFossilCooper() as MiniCooper.BaseMiniCooper ??
miniCooper.GetHybridCooper() as MiniCooper.BaseMiniCooper;
```

Her er hvordan vi "Referencer" vores klasser, og loader alt ind i en flot række som vist, hvor noget default bootstrap jeg har spillet med, er det visuelle.

Sprint review

I slutningen af et sprint, vil teamet have et sprint review møde. I dette møde vil teamet tilpasse vores product backlog hvis nødvendigt, samt vurdere hvad der er afsluttet i sprintet sammen med interessenter. Deltagerne af dette møde vil også diskutere den næste funktionalitet som kan blive færdig.

I dette sprint review afholdes med udviklingsteamet sammen med scrum master. Vi har et planlagt møde med Product owneren / kunden Peter fredag d. 6/12, hvor vi vil vise ham hvad vi har lavet, og få hans feedback på det.

Vi startede sprintet med at holde et sprint planning møde, hvor vi kortlagde vores arbejde. Her udarbejdede vi vores sprint goal, ud fra den user story, som vi valgte at arbejde med i dette sprint. Vi lavede derefter et "Github Projects"/sprint backlog, for at danne et overblik over sprintet, hvor vi bl.a. dannede tasks.

Teamet har været gode til at opdatere "Github Projects" løbende med, hvad de enkelte i teamet starter eller bliver færdige med i en task. "Github Projects" har samtidigt givet teamet et godt overblik over, hvad de andre på teamet arbejder med, og hvad der mangler at blive lavet i sprintet.

Vi lavede derefter et database-diagram. Dette hjælper os med at skabe overblik og enighed omkring, hvordan vores database skal se ud - opstillet på sådan en måde, at både holdet og interessenter har mulighed for at danne et overblik over databasens relationer.

Vi endte med to versioner, da vi hurtigt fandt ud af, at vores første version² var mere kompliceret end hvad der var nødvendigt. Sammen løste vi dette i fællesskab ved at forbedre og simplificere opbygningen af modellen. Dette var også nemmere, da vores forståelse for måden diagrammet skulle opbygges for at gøre den overskuelig. Det medførte dermed til sidst vores endelige version.

Eftersom at vi er færdige med database-diagrammet, fik udviklingsteamet lavet en database med udgangspunkt i vores database-diagram.

Derfor valgte vi at oprette det i "Neon.Tech", som vi delte med hele udviklingsgruppen. For at dele arbejdet op imellem os, fik vi oprettet branches, som er navngivet efter alle udviklingsgruppens navne.

Ud fra modellerne fik vi oprettet databasen. Dernæst oprettede vi alle klasser ud fra de klasser, som vi havde vist i klassediagrammet.

Sprint retrospective

Vi startede sprintet med at afholdte et sprint planning møde, hvor vi kortlagde vores arbejde. Her udarbejdede vi vores sprint goal, ud fra de tasks, som vi valgte at arbejde med i dette sprint. Vi lavede derefter et "Github Projects"/sprint backlog, for at danne et overblik over sprintet.

Teamet har været gode til at opdatere "Github Projects" løbende med, hvad de enkelte i teamet starter eller bliver færdige med i en task. "Github Projects" har samtidigt givet teamet et godt overblik over, hvad de andre på teamet arbejder med, og hvad der mangler at blive lavet i sprintet.

Vi lavede derefter et database-diagram. Dette hjælper os med at skabe overblik og enighed omkring, hvordan vores database skal se ud - opstillet på sådan en måde, at holdet har mulighed for at danne et overblik over databasens relationer.

² Bilag 1 - Gammel database-diagram

Vi endte med to versioner, da vi hurtigt fandt ud af, at vores første version var mere kompliceret end hvad der var nødvendigt. Sammen løste vi dette i fællesskab ved at forbedre og simplificere opbygningen af diagrammet. Dette var også nemmere, da vores forståelse for måden diagrammet skulle opbygges for at gøre den overskuelig. Det medførte dermed til sidst vores endelige version.

Eftersom at vi er færdige med database-diagrammet, fik udviklingsteamet lavet en database med udgangspunkt i vores database-diagram.

Derfor valgte vi at oprette det i "Neon.Tech", som vi delte med hele udviklingsgruppen. For at dele arbejdet op imellem os, fik vi oprettet branches, som er navngivet efter alle udviklingsgruppens navne.

Ud fra modellen fik vi oprettet databasen. Dernæst oprettede vi alle klasser ud fra de klasser, som vi havde vist i klassesdiagrammet.

Vi havde et møde, hvor vi planlagde vores MVP (Minimum Viable Product). Ideen med et MVP, er at få det mest basale på plads, så vi alle sammen har en klar idé om, hvad vi skal gøre for at kunne bestå. Dette giver os en klar ide om hvad vi skal, samtidigt med at vi kan sætte en masse andre ideer og bekymringer væk, så vi kan fokusere på det vigtige. Når vi så har fået klargjort det minimale, kan vi slappe mere af, samtidigt med at vi gør vores projekt endnu bedre.

Hvad vi kom frem til var 5 kriterier:

- Brugeren skal have mulighed for at se alle de listede Coopere.
- Brugeren skal kunne få en detaljeret oplysningsside i forhold til en bil.
- Brugeren skal kunne oprette/logge ind.
- Brugeren skal have CRUD muligheder i forhold til deres Coopere.
- Man skal kunne relatere en Cooper til en bruger. Skal inkludere en "relativ" lokation i forhold til hvor man kan få bilen.

Vi har også udviklet en minimalistisk front-end, hvor man ser alt info om biler i en simpel og praktisk liste, som taler direkte med databasen uden fejl.

Der er stadig meget tilbage at lave, visuelt-mæssigt, men uden for lidt ekstra pynt, så er den brugbar, og har et design man genkender fra andre lignende sider.

Front-end var bygget igen fra grunden af, siden det sidste bare var en "Demonstrator" med randomly generated flavortext, og manuelt indsatte billeder, men er siden blevet helt ombygget til at have et modular, og standardiseret layout der kan bruges alle steder i siden, som MainLayout, og Navbar header.

Andet sprint

Første Sprint var fra d. 9/12 - 13/12 2024.

Sprint planning

Under planlægningen af Sprint 2 stod det klart, at vi ikke havde udnyttet "Github Projects" optimalt i det foregående sprint. Dette gjorde det udfordrende at skabe et præcist overblik over, hvilke opgaver der var blevet gennemført, og hvilke der stadig manglede. For at forbedre dette prioriterede vi i sprint planning at strukturere vores backlog og tasks mere grundigt. Vi inddelte arbejdet i mindre opgaver og lagde en klarere plan for sprintets mål. Vi overvurderede den tid, der var til rådighed, hvilket gjorde det nødvendigt at justere vores ambitionsniveau senere i sprintet.

Sprint Backlog

Vores sprint backlog blev udfyldt med de opgaver, vi så som vigtige for at nå sprintets mål, herunder videreudvikling af funktioner og implementering af ny kode. Vi besluttede at navngivning af tasks i backloggen blev forvirrende, da der var mange tasks, kunne man hurtigt tage fejl af hvilke tasks som allerede er oprettet og i stedet komme til at tilføje tasks som allerede var til stede. Det afgjorde at vi besluttede os for at ændre måden, hvor vi i resten af projektet skulle navngive tasks. De skulle navngives først med den enkelte side på hjemmesiden. Efterfulgt af det skulle der være en parentes med tekst som beskrev om det enten er "Funktionel", "Styling" eller "Validering".

Desværre opstod der udfordringer med at holde styr på tasks i "Github Projects," da teamet ikke konsekvent opdaterede status på opgaverne. Dette resulterede i en manglende koordinering og en fejlmargen i vores vurdering af, hvor langt vi var med sprintets fremdrift. Det førte til en dominoeffekt, hvor vi mod slutningen af sprintet måtte fjerne flere planlagte opgaver fra backloggen for at sikre levering af minimumsproduktet (MVP).

Sprint review

Ved sprint review med product owner / kunde Peter var det tydeligt, at teamet havde arbejdet med en udmærket fremdrift, men alligevel ikke nåede alle de planlagte tasks. En årsag til dette var, at tidsforståelsen for enkelte tasks og arbejdsfordelingen ikke var tilfredsstillende nok. Ud over det skulle vi have været bedre til at kommunikere med hinanden om at løse de kode problemer som endte med at fylde hele sprintet. Så kunne problemerne i koden have været taget hånd om tidligere, og der ville have været mere overskud til at løse de opgaver, som hvert teammedlem er bedre til at løse. Manglende brug af "Github Projects" skabte også usikkerhed for den præcise status af flere tasks i løbet af sprintet. Til trods for udfordringerne lykkedes det os at levere mange af de vigtigste funktioner til hjemmesiden. Efter den feedback vi fik fra product owner / kunde Peter, besluttede vi os for at omprioritere de resterende tasks samt tilføje tasks til sprint 3

Sprint retrospective

I retrospektiv reflekterede teamet over, hvordan vi kunne forbedre vores workflow. Vi erkendte, at en mere regelmæssig brug af "Github Projects" er nødvendigt for at skabe overblik, så vi ikke skulle bruge for meget tid på at snakke om, hvor langt vi er nået. Der var også forvirring om, hvor langt alle teammedlemmer er nået. Vi blev enige om, at vi skal være

bedre til at opdatere alle status for task, så alle på holdet altid kan se, hvor langt vi er. Derudover besluttede vi at være mere realistiske i vores estimering af tidsforbrug, da vi undervurderede udfordringen og arbejdstid af nogle tasks i dette sprint. Det er meningen at disse erkendelser skal forhindre os i at komme bagud i tidsplanen i Sprint 3.

Tredje sprint

Første Sprint var fra d. 16/12 - 18/12 2024.

Sprint planning

I Sprint 3 fokuserede teamet på at planlægge implementeringen af de mest essentielle funktioner for projektet. Målet var at skabe en Minimum Viable Product (MVP), hvor brugeren kunne se alle listede Cooperer, få adgang til en detaljeret oplysningsside om hver bil, registrere og logge ind samt udføre CRUD-operationer på deres egne Cooperer. Derudover skulle der etableres en relation mellem brugere og biler, inklusive information om en relativ lokation for bilerne. Som en mulig udvidelse blev det overvejet, at brugerne også skulle kunne købe en Cooper, men denne funktion blev ikke prioriteret i første omgang.

Sprint Backlog

I løbet af sprinten blev backloggen organiseret omkring flest styling tasks, som omfattede visningen af en liste over alle Cooperer, implementeringen af en detaljeret informationsside for hver bil, oprettelsen af registrerings- og login-funktionalitet, CRUD-operationer for brugernes egne Cooperer samt relationen mellem brugere og biler, herunder lokationsinformation. Tasks blev organiseret i "GitHub Projects", hvor de blev tildelt ansvarlige personer, planlagt til specifikke sprints og prioriteret efter vigtighed. Kommunikation foregik via Teams og fysisk fremmøde, hvilket sikrede, at teamet var opdateret og kunne samarbejde.

Klassediagram



Sprint review

Ved afslutningen af sprinten blev de fleste funktionelle krav opfyldt, hvilket resulterede i en næsten fuldført MVP. Brugere kan nu se en liste over alle Cooperer, få detaljer om hver bil, registrere og logge ind samt administrere deres egne Cooperer. Så relation mellem brugere og biler implementeret. Funktionen med at købe en Cooper blev dog ikke implementeret, da teamet vurderede det som en mindre prioritet for, hvad vi kunne nå i sprintet.

Sprint retrospective

I dette retrospektiv reflekterede teamet over udfordringer og hvad som kunne forbedres. Det som fungerer bedre i dette sprint modsat andet sprint var den navngivning af tasks i backloggen, som tidligere var for bredt og svære at arbejde med. Dette blev løst ved at

opdele opgaverne i mindre dele, der fokuserede på enten funktionalitet eller styling. Den selvstændige arbejdsmetode skabte også uvidenhed omkring, hvilke udfordringer hver især har. Derfor var der behov for ofte at kommunikere sammen. Frem mod slutningen af sprinten blev kommunikationen dog markant forbedret vores samarbejde.

Fremadrettet ville vi have ønsket at løse flere forbedringer til hjemmesiden. Bl.a. muligheden for at redigere en Cooper direkte fra dashboardet, implementering af thumbnails for forsidebilleder, en fysisk repræsentation af klasse- og databasestrukturer til reference, udviklingen af en chatfunktion mellem sælger og køber samt forbedring af sikkerheden for registrerings- og login ved hjælp af kryptering.

Bilag

Bilag 1 - Gammel database-diagram - 1. sprint

