

Tanzania Eksperten

Rapport – Systemudvikling 2

UCL Erhvervsakademi og Professionshøjskole

DMVF241 – 3. semester

Forår 2025



Skrevet af Karina, Amjad, Thomas og Rene

Indholdsfortegnelse

Indledning	3
Problemformulering	3
Afgrænsning	4
Metode.....	5
Analyse	6
Kravanalyse og systemudvikling	6
Domænemodel:.....	7
Klasse diagram:	8
Entity-Relations diagram.	9
Brief Use Case	9
Casual Use Case	9
Fully Dressed Use Case.....	10
Valg og brug af procesmodel	12
SCRUM	12
Projektstyring.....	13
Teststrategi og kvalitetssikring	13
Prototype præsentation.....	15
Diskussion.....	17
Konklusion	18
Perspektivering	18
Litteraturliste	19
Bilag.....	19

Indledning

TANA er et webapplikationsprojekt udviklet til intern brug af Tanzania Eksperten, en virksomhed der tilbyder ture til Tanzania. Den er designet med en brugervenlig grænseflade, der sikrer en problemfri arbejdsgang i virksomheden.

Applikationens formål er at håndtere oprettelse af rejseplaner, administration af ture, fakturaer og kunder, hurtigt oprettelse af rejseskabeloner med understøttelse af ændring af skrifttyper og farver og afsendelse af rejseplaner til kunder via e-mail til godkendelse.

Brugeren kan oprette en rejseplan med fuld kontrol, for eksempel kan han vælge antal dage, pris, indkvartering, aktiviteter, vælge flyselskab og sende detaljerne til klienten som en PDF-fil. Muligheden for at tilpasse branding er også tilgængelig. Applikationen giver også mulighed for at oprette klienter, der indeholder det fulde navn og kontaktoplysninger som telefonnummer, e-mail, adresse og klientens status, uanset om den er aktiv eller inaktiv.

Denne rapport dokumenterer faserne i systemudviklingen, kravanalysen med brugerhistorier og prioriteringen.

Vores processmodel for udviklingen af systemet var SCRUM med sprintplanlægning og brug af GitHub-projekter til at styre opgaver.

Systemarkitekturen er baseret på principperne om Clean Architecture og domænefokus, mens bruger fronten var oprettet via Blazors framework

Test og kvalitetssikring ved hjælp af xUnit blev anvendt til test af funktionelle krav – såsom oprettelse af rejseplaner, administration af kunder og fakturering – og ikke-funktionelle krav såsom sikkerhed til kryptering af følsomme kundeoplysninger, brugervenlighed og opetid, i samarbejde med produktejeren.

Problemformulering

Tanzania Eksperten kæmper med hurtigt at oprette rejseplaner og generere PDF-tilbud ved hjælp af genbrugelige skabeloner. er det vigtigt for rejsebureauer nemt at kunne ændre layout, farver og skrifttyper og send professionelt designede tilbud til kunder.

Peter har også helt klart brug for, at flere brugere kan arbejde samtidigt på systemet uden risiko for databas. Peter bemærkede også, at det skal være muligt at redigere, oprette og slette klienter, hvilket sikrer, at disse oplysninger er beskyttet, mens de gemmes i databasen eller overføres.

Men hvordan skal vi på 6 uger bygge dette system, der opfylder TANA krav, fra rejseplanlægning til kundeforsendelse på en organiseret måde, med fokus på flerbrugerfunktionalitet og datasikkerhed?

For at besvare dette spørgsmål, vil vi fokusere på følgende underspørgsmål:

1. Hvem og hvilke brugerroller og domæneentiteter skal bruges til at bygge dette system?
2. Hvordan prioriteres og organiseres kravene fra produktejeren?
3. Hvilken udviklingsmetode og systemarkitektur skal anvendes for systemet for at sikre en fleksibel og testbar løsning?
4. Hvordan kan vi designe og implementere en genbrugelig skabelon og sende den til klienten?
5. Hvordan kan kundedata beskyttes i henhold til den generelle forordning om databeskyttelse (GDPR), og hvordan kan brugerens rolle bestemmes ved login?

Afgrænsning

Vi har arbejdet tæt sammen med Product Owneren (Peter) i vores Tanzania projekt, for at identificere de vigtigste arbejdsgange og behov i virksomheden. Vi har valgt at fokusere på oprettelse og håndtering af rejseplaner, administration af standardture og tilpasning af rejser, kundeoversigt og kundeoplysninger, oprettelse af PDF'er til deling med kunderne, og oprettelse af faktura. Vi har ikke haft så meget fokus på automatisk email-afsendelse, kortintegration med f.eks. Google Maps, betalingsløsninger eller selvbetjeningsløsninger for kunderne.

Vores testarbejde har været begrænset pga. Projektets tidsramme.

Vi har anvendt TDD (Test Driven Development) i den tidligste fase, i sprint 0.

Applikationslaget er opbygget med interfaces, hvilket gør det muligt at mocke og isolere test. Vi valgte at bruge xUnit til at skrive unit tests. Vi har undladt at lave integrationstests desværre, det havde vi ikke tid til. Vores frontend er kun testet manuelt gennem UI-interaktion i browseren.

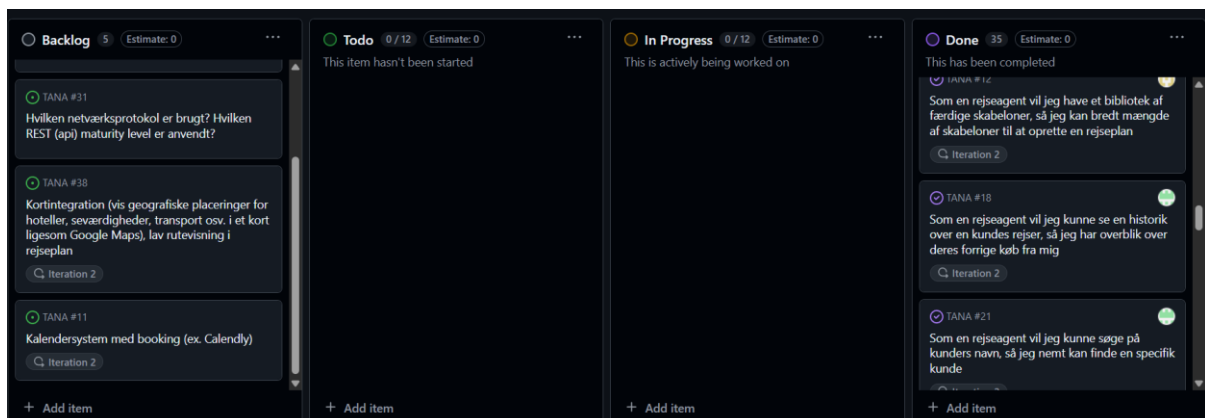
Vi nåede ikke at lave projektet til en mobilapp, pga. Tidsbegrænsninger, såvel som at kunder heller ingen adgang har til systemet, der er ikke blevet lavet en betalingservice, og der er en begrænset kalenderintegration.

Dvs. At vi kun har en desktop webapplikation oppe at køre, kun rejsebureauets medarbejdere og administratorer kan logge på og har adgang til systemet, brugerne kan ikke synkronisere deres rejser med google eller outlook, og at systemet ikke understøtter betaling for rejserne.

Nogle af de begrænsninger vi har i projektet er, at systemet skulle bygges i Blazor, .NET og MSSQL. At vores kode skulle følge Test Driven Development og SOLID-principperne. Kun autoriserede brugere, altså administratorer og ansatte, skulle have adgang til nogle bestemte funktioner. Vores projekt skulle hostes lokalt med en lokal API. At vi skulle kunne generere PDF'er med QuestPDF. Vi ville lave templates som PDF'er. Vi skulle følge og kode med Clean Architecture. Der skulle kun være begrænset data, så kun enkelte felter med pris, dage, beskrivelse og navn for ture og rejseplaner. Og at vi havde en tidsbegrænsning, hvor projektet skulle afleveres indenfor en bestemt tid, inden semesteret er overstået, og inden de mundtlige eksamener.

Metode

Vi brugte SCRUM og Kanban board i Github Projects. Vores Kanban board var opdelt mellem vores backlog, To Do, Doing og Done. Vores systemudviklings proces var agile igennem SCRUM. Vi opdelte vores arbejdsproces efter sprints, som hvert var en uges varighed. Herfra opdelte vi opgaverne i sprintet mellem os.



Vores SCRUM backlog indeholdt user-stories for alle de funktioner som vi ville have at programmet indeholdt, plus et par ekstra reminders for os, som var relevant for det overordnet projekt.

Vi brugte GitHub-Projects til at oprette og vedholde vores Kanban board, og brugte Discord som platform til at planlægge sprints, uddele user-stories til de individuelle medlemmer og general kommunikation. Det var også i GitHub hvor vi oprettede vores projekt som et repository. Det gav alle i gruppen nem adgang til koden. Via branching, lod det også os arbejde asynkront på forskellige funktioner. Herfra kunne vi merge vores forskellige branches sammen for at skabe vores endeligt program.

SCRUM-projektet opdeles i sprints, for os en uges varighed. Et SCRUM-team består af en Product Owner (Peter), SCRUM-master (René) og udviklingsteam (Amjad, Thomas, Karina).

Hvert sprint starter med en sprint planning og afsluttes med en sprint review og sprint retrospective. Sprint planning er hvor vi medlemmer af projektet, afgør og siger hvad vi kan levere til kommende sprint og få gjort færdigt.

Sprint review afholdes efter hvert sprint, og det er at vise hvad vi har fået gjort færdigt i sprintet, og samle feedback til hvad der kan gøres bedre til næste sprint.

I sprint retrospectivet snakker vi i SCRUM-teamet, om hvad der gik godt og hvad der gik dårligt, det var f.eks. også sammen med product owneren (Peter), hvor vi fik feedback på hvad der skulle gøres bedre, og hvad der var godt lavet.

Vi havde daily standups hver dag eller hver anden dag på Discord, hvor vi bare skrev hvad vi ville gå i gang med for dagen, og hvad vi var blevet færdige med siden sidste sprint. Det var også her hvor vi tilgæede hinandens hjælp, hvis vi havde brug for det.

Til vores arkitektur valgte vi at bruge Clean Architecture. Vi oprettede de relevante libraries/projekter til at oprette systemet efter den metode, og oprettede derefter klasserne, metoder, services m.m. efter behov.

Analyse

Kravanalyse og systemudvikling

En af de vigtigste funktionelle krav i systemet er brugerhåndteringen, hvor vi har et login for brugere (ansatte og administratorer), en rollebaseret adgang, hvor kun administratorer kan oprette og redigere ansatte, og en mulighed for at ændre e-mailadresse for PDF-afsendelse både for administratorer og ansatte.

En anden vigtig funktionel krav er rejseplanlægningen, hvor vi kan oprette standardture med navn, pris, dage og beskrivelse. Vi kan også oprette rejseplaner ved at sammensætte én eller flere standardture. Vi kan også oprette specialisering af ture ved at ændre pris eller dage. Og så er der tilføjelse af begivenheder som f.eks. byvandring, cykeltur og tur i søen til en rejseplan.

En tredje vigtig funktionel krav er kundeoversigt, hvor vi kan oprette og vise kundedata. Her kan statussen for individuelle kunde også ses, så det er muligt at for overblik over hvor mange og hvor langt de kunder er i bookingsprocessen.

En fjerde vigtig funktionel krav er fakturering og PDF-generering, hvor vi kan automatisk generere PDF'er med oplysninger om rejseplanen, der er mulighed for at tilpasse layoutet af PDF'erne med skabeloner, og PDF'erne kan sendes til en mail.

En femte vigtig funktionel krav er rejseoversigten, hvor vi har en oversigt over oprettede rejseplaner med tilknyttede ture og kunder.

De vigtigste ikke-funktionelle krav er brugervenlighed, sikkerhed og adgangskontrol, vedligeholdelsesvenlig arkitektur, ydeevne, testbarhed og tilgængelighed.

Systemet er intuitiv og let at bruge for rejsebureauets medarbejdere, og der er fokus på en simpel brugerflade i Blazor. Kun autoriserede brugere på få adgang til systemet vha. Et login og rollebaseret adgangsstyring er implementeret. Projektet/systemet er bygget op med Clean Architecture og Domain Driven Design for at adskille mellem domænelogikken, applikationen og præsentationen. Vi har brugt interfaces og dependency injection for testbarhed og fleksibilitet. Projektet er lavet med .NET 8 og kan hostes i en Docker container for nem deployment. Vi har brugt Test Driven Development og xUnit for at teste kernefunktionaliteten, hvor vi også har gjort brug af moq og mocking. Systemet skulle helst gerne kunne køre stabilt internt uden at gå ned.

Vi har som udviklingsteam identificeret de funktionelle krav gennem samtaler med vejlederen og ud fra projektets formål og løbende prioritering under SCRUM-processen.

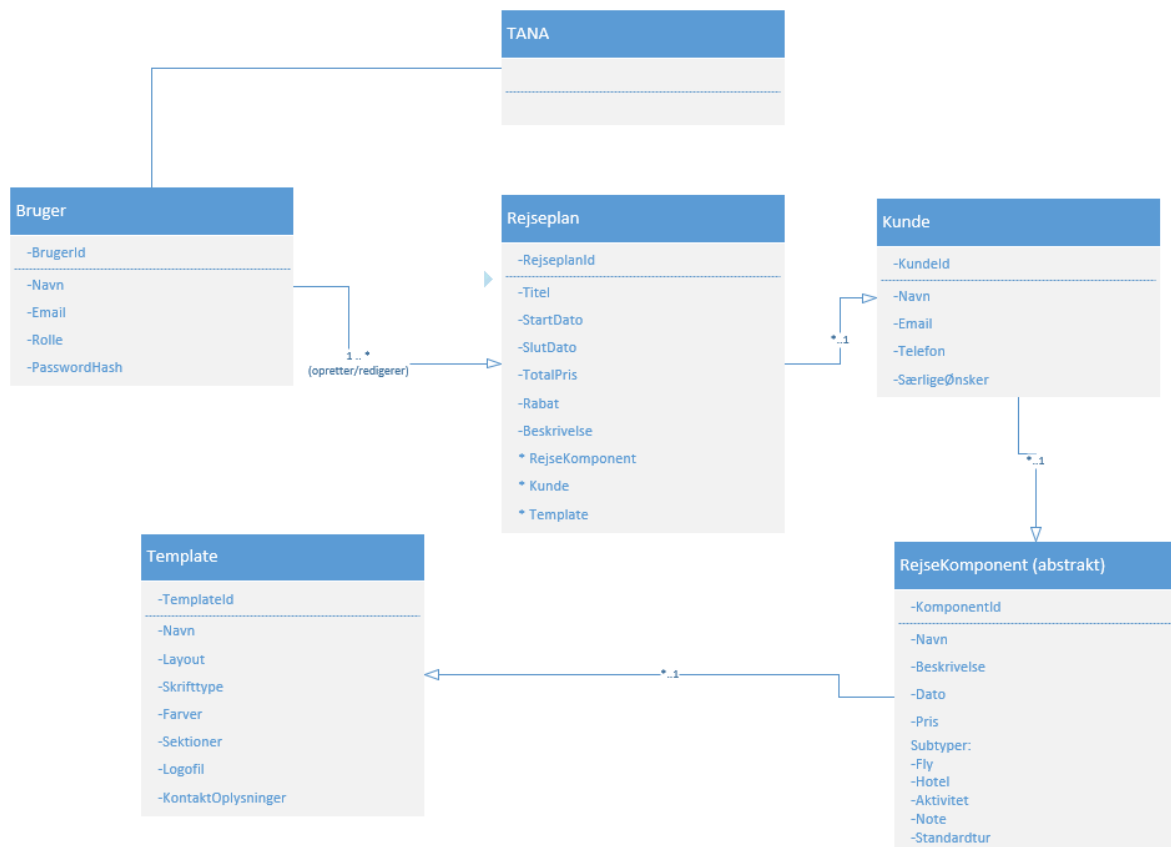
Systemet lever op til de grundlæggende krav, der er enkelte funktioner som selvbetjening for kunder og integrationstest, som er bevidst udeladt, og derfor uden for projektets scope.

Før vi gik i gang med udviklingen af vores projekt, lavede vi en række diagrammer for at få overblik over casen. Dette indeholdt en domain model, et klassediagram og et Entity-Relations diagram. Vi lavede også flere use-cases, for at indsnævre detaljerne omkring de individuelle funktioner i systemet.

Domænemodel:

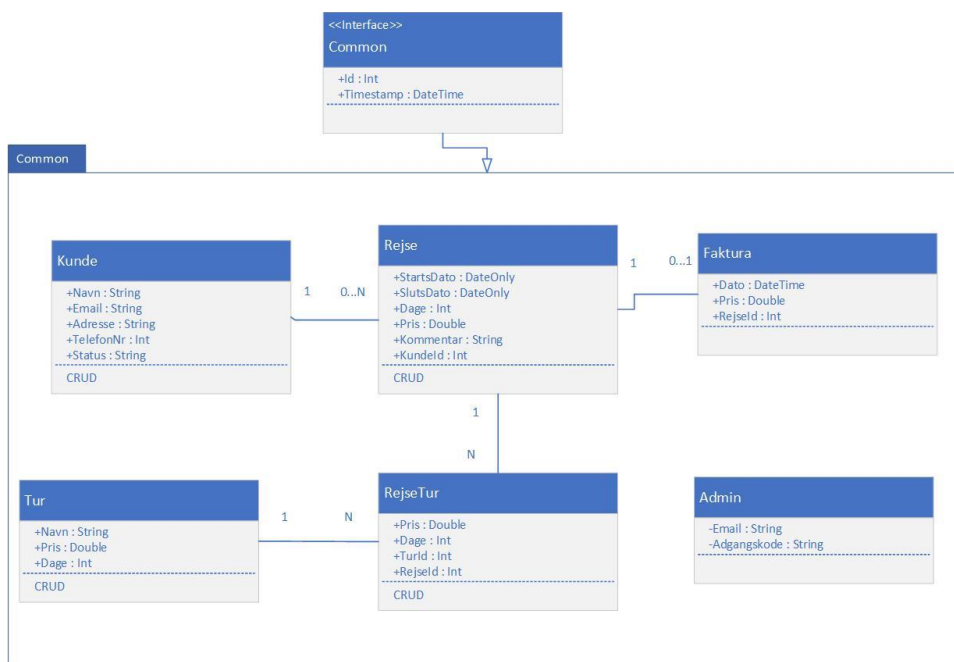
Brugeren er rejsebureauets medarbejdere. De kan logge ind og administrere rejseplaner og kunder. En rejseplan oprettes og redigeres af brugere og knyttes til en eller flere kunder. De indeholder en samling af rejsekomponenter. Kunderne har personlige oplysninger og særlige ønsker. De kan knyttes til én eller flere rejseplaner.

Rejsekomponent er en abstrakt entitet, som kan være en specifik aktivitet, hotelbooking, flyrejse, note eller en standardtur. Template bruges til at generere PDF'er med branding, layout og indhold. De er tilknyttet rejseplaner.



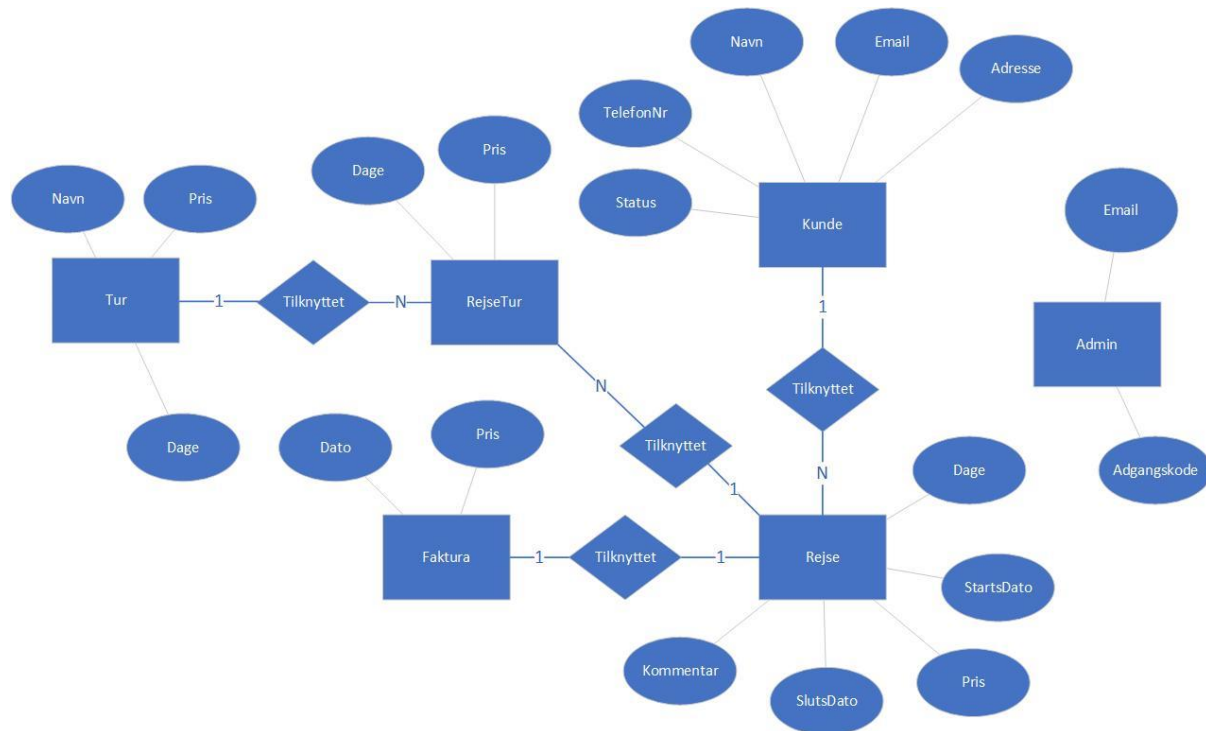
Klasse diagram:

For Klasse diagrammet fokuseret vi på kernerollerne som vores domæne lager vil opstå af. Da alle af vores klasser har brug for et Id og en Version timestamp implementerede vi et IEntity interface.



Entity-Relations diagram.

Vores EF-diagram fungerede som skabelonen for vores database. Den indeholder de attributter for alle de klasser vi havde brug for at gemme information om i vores database. RejseTur klassen fungerer som en forbindelse mellem Rejse og Tur klassen. Da de to klasser har en mange-til-mange relation, kræver de denne mellemliggende klasse.



Brief Use Case

Titel: Opsætning af booking og plan for betaling af Marangu Tur

Feature: Booking og betaling

Beskrivelse:

Når en kunde har bekræftet deres 9 dages Marangu tur, så kan en rejsebureau medarbejder lave en booking. Systemet giver medarbejderen muligheden for at lave en faktura, hvor de selv kan opsætte forfaldsdatoer, rater af betaling samt sende automatiske betalingspåmindelser. Systemet sender betalingslinks via Stripe eller PayPal.

Casual Use Case

Titel: Udskiftning af turaktiviteter

Feature: Ændringer af tur samt versionsstyring

Beskrivelse:

En kunde modtager en rejseplan for deres 12 dages Lemosho tur. Den modtages som PDF via email. Kunden svarer tilbage på mailen at alt er perfekt, men de ønsker dog at fjerne Lava Tower og tage en ekstra nat i Shira Camp.

Bureaumedarbejderen bruger systemets drag and drop løsning, til at fjerne Lava Tower aktiviteten og kan tilføje en ekstra nat i Shira Camp. Systemet gemmer den nye version, samt logger ændringen. Den tidligere version er stadig tilgængelig, og kan gendannes hvis nødvendigt.

Fully Dressed Use Case

Titel: Oprettelse af skræddersyet rejseplan ved hjælp af skabeloner og moduler

Primær aktør: Rejsebureaumedarbejder

Mål: Oprette en personlig og gennemført rejseplan ved hjælp af skabeloner og moduler, tilpasset efter kundes behov, med evt. inspiration fra allerede oprettet ture.

Omfang: TANE Rejseplanssystem

Level: Bruger-mål

Interessenter og interesser:

Bureaumedarbejder	Ønsker at lave detaljerede rejseplaner, hurtigt og effektivt med skabeloner og moduler.
Kunde	Vil have en personlig og struktureret rejseplan med præcise oplysninger.
Product-owner	Ønsker skalerbare og effektive systemer, som sikrer hurtig udvikling af rejseplaner.

Forudsætninger:

- Bureaumedarbejderen er logget ind i systemet.
- Skabeloner fra tidligere ture er tilgængelige.
- Moduler i form af aktiviteter er tilgængelige i databasen.

Resultat:

En komplet rejseplan, der passer kundes behov, med oplysninger om rejsedage, aktiviteter, højder og camps. Planen kan downloades som PDF.

Hovedforløb:

1. Medarbejder vælger "Opret ny rejseplan"
2. Medarbejder vælger en skabelon som "12 dages Lemosho tur"
3. Systemet indlæser en kladde med forskellige aktivitetsmoduler og ruter.
4. Medarbejderen tilpasser rejseplanen.
 - a. Justerer antal dage
 - b. Udskifter aktiviteter efter kundes behov
 - c. Beskrivelser og indhold opdateres i rejseplanen
5. Systemet tjekker for logistiske fejl
6. Medarbejderen forhåndsviser og genererer en PDF
7. PDF deles til kunde via email og med link
8. Systemet gemmer den ændrede rejseplan med mulighed for versionsstyring

Alternative forløb:

4a. Ekstra oplevelser

1. Kunden ønsker ekstra aktiviteter tilføjet
2. Medarbejderen udvider rejseplanen med modulerne

5a. Overlappende aktiviteter

1. Systemet advarer om overlap i rejseplanen
2. Medarbejder korrigerer dette

8a. Kunde ønsker ændring

1. Kunde ønsker større ændring senere
2. Medarbejder kopierer den nuværende rejseplan
3. Laver den nye version tilpasset kundens ønsker

Særlige krav:

- Alle destinationer skal inkludere højde
- Systemet skal automatisk indsætte information om vejr og sundhed ved højder over 3.000m
- Skabeloner må ikke slettes, men klones og tilpasses
- Ændringshistorik skal vise redaktør, dato og beskrivelse af ændringen

Hyppighed:

Høj - da det anvendes dagligt af medarbejdere.

Valg og brug af procesmodel

Vi valgte at bruge SCRUM da det var en agil udviklingsmetode vi som et team allerede havde godt kendskab til. Det gjorde det nemmere for os at håndtere arbejdsprocessen, da vi ikke behøvede at bøjede med en udviklingsmetode som vores team ikke havde prøvet før.

Vi valgte ikke at bruge XP, fordi flere af vores medlemmer hovedsageligt arbejdede virtuelt og asynkront. Det ville derfor være svært at implementere, da vi ikke ville kunne have den store mængde af kommunikation og oversigt som XP kræver.

For mange af de samme grunde var par programmering ikke velegnet for os. Vi så det også som mere besvær end det var værd. Fordi vores system ikke er særlig stor, vil de optimeringer som par programmering bringer, ikke være tiden værd for os i denne omstændighed

SCRUM

Vi startede vores arbejdsproces med et uofficielt sprint 0. Det var her hvor vi arbejdede med alle de ikke-programmering opgaver vi skulle have lavet. Dette inkluderede diagrammer, vores GitHub repository, Kanban board, arkitekturen for projektet og oprettelsen af vores database. Fordi vi stadig havde normale timer under denne periode, varede vores sprint 0 to uger. Længere en den normale varighed for et sprint i vores projekt af en uge.

De næste to sprints fokuserede vi på kernefunktionerne af vores projekt. Efter DDD og en Api first fremgang, oprettede vi først de klasser, services, repositories og Api'er vi havde brug for til at oprette vores funktioner. Herefter arbejdede vi med pdf-generering, skabeloner og kunde/rejse oversigt, da de er de vigtigste funktioner i programmet.

Sprint 3 finjusterede vi vores sider og tilføjede de mere "Quality of life" funktioner for programmet, såsom søgning på kunder. Det var også i dette sprint vi designede størstedelen af ui'et for programmet.

I sprint 4 puttede vi de sidste pynt på programmet, lavede exploratory tests og implementerede Docker filer. Efter det var færdigt, begyndte vi med at skrive på de to rapporter, hvilket tog det meste af tiden af dette sprint.

Projektstyring

For at styre projektet gjorde vi brug af GitHub Projects og dets funktioner. GitHub Projects var praktisk for os, da vi allerede brugte GitHub til at hoste vores repository. Vi kunne derfor forbinde vores repository direkte til vores Project.

Vi indsatte alle vores User Stories ind på et Board. Her kunne vi markere hvert individuelt User Story som enten i Backlogget, en del af vores nuværende sprint (Todo), noget vi arbejder nuværende med (In Progress), eller som fuldført (Done). I GitHub Projects er det også muligt at markere hvilket medlem af vores gruppe arbejdede med hvert af User Stories og også på hvilken iteration af programmet det skulle fuldføres. Ved brug af de iterationer oprettede GitHub automatisk en roadmap, som viste hvilke User Stories vi fuldførte i hvilke sprints.

Det ville også være muligt at sætte start og slut datoer på dem for at indramme iterationerne yderligere, men det gjorde vi ikke brug af i dette tilfælde.

Vores Kanban board hjalp os med at holde overblik både over projektet i det hele taget, og over de individuelle sprints.

For at forsikre om at alle i teamet fik lavet deres tildelt opgaver holdt vi periodiske møder. Disse møder forgik ved starten ud slutningen af hvert sprint. Vi holder dem hovedsageligt fysisk, men på grund af nogle ydre omstændigheder blev vi også nødt til at holde nogen af dem virtuelt over Discord. I møderne starten af sprintet diskuterede vi projektets nuværende tilstand og hvilke funktioner der ville være bedst at arbejde på næste sprint. Vi udelte opgaverne efter først til mølle-princippet, så de medlemmer med en præference/styrke for et specifikt emne kunne vælge det, og dem uden kunne tage de opgaver der var tilovers.

Møderne i slutningen af sprints var hvor vi fremviste de funktioner vi havde lavet over det nuværende sprint og mere bredt diskuterede vores systemets nuværende tilstand. Den fysiske tilstedeværelse gjorde det også nemmere for os at hjælpe hinanden og fikse bugs.

Teststrategi og kvalitetssikring

Vores gruppe primære test-metode var exploratory-testing blandet med Unit testing. Ved exploratory testing, indsatte vi dummy data i vores databasen og kørte programmet i Visual Studio. Da vi direkte afprøvede vores funktioner, gjorde det det nemt at finde de mest alvorlige bugs i vores program, de bugs der ville have direkte indflydelse på brugervenligheden af systemet. Derimod med Unit testing kunne vi teste mere smallere kodesnip. Vi brugte f.eks. xUnit testing til TDD, som var det første vi gjorde i Sprint 0. Her testede vi om når vi opretter en bruger, om brugeren så har et tilgængeligt brugernavn.

Vi fik nogle gange et par fejl i testene, men det var ikke noget vi ikke kunne overkomme, ved at refaktorere eller ændre lidt af software-koden, for at passe til testene. Kun en meget lille procentdel af koden er blevet testet, nemlig kun `CreateUserAsync_ShouldCreateUser_WhenUsernameIsAvailable`.

```
public class UserServiceTests
{
    private readonly Mock<IUserRepository> _userRepositoryMock;
    private readonly IUserService _userService;

    0 references
    public UserServiceTests()
    {
        _userRepositoryMock = new Mock<IUserRepository>();
        _userService = new UserService(_userRepositoryMock.Object);
    }

    [Fact]
    0 references
    public async Task CreateUserAsync_ShouldCreateUser_WhenUsernameIsAvailable()
    {
        // Arrange
        var username = "testuser";
        var password = "password123";

        _userRepositoryMock.Setup(r => r.GetByUsernameAsync(username))
            .ReturnsAsync((User)null);

        _userRepositoryMock.Setup(r => r.AddAsync(It.IsAny<User>()))
            .ReturnsAsync((User u) => u);

        // Act
        var result = await _userService.CreateUserAsync(username, password);

        // Assert
        Assert.NotNull(result);
        Assert.Equal(username, result.Username);
    }
}
```

Vi mødte også med produktejeren hvor vi fremviste systemets nuværende tilstand. Derfra fik vi feedback på vores system fra en lægmands synspunkt. Hans feedback hjalp også med at styre direktionen af systemet, ved at udnævne funktioner af større betydning.

En måde vi kunne skabe direktion vil være igennem metrikker. Metrikker er målbare data, som bruges til at måle og vurdere de forskellige aspekter af softwareudviklingen og tests.

Typiske metrikker vil inkludere måling i et sprint af linjer kode skreven og nummer af defekter fundet i testning. Disse målinger fik vi dog desværre ikke målt i vores projekt. For at finde defekter brugte vi en mindre metodisk tilgang. Det vil være noget vi vil gøre anderledes i fremtidige projekter.

Prototype præsentation.

Opret Rejseplan, gør at du kan oprette rejseplaner, ved enten at inkludere en tur/standard tur, eller lade vær, du kan ændre pris og antal dage og tilføje en beskrivelse. Du kan også slette allerede eksisterende rejseplaner.

TANA.Web

Home

Counter

Weather

Modify Email

Send Rejseplan

Opret Rejseplan

Rejser

Standard ture

login

Opret Rejseplan

Navn: Beskrivelse: Pris: Dage:

Vælg Ture:

☐ Guidet byvandring

☐ Cykeltur

☐ Svømmetur i søen

Samlet pris: 0 kr.

Samlet antal dage: 0

Opret Rejseplan

Allerede oprettede rejseplaner

- Tanzania**
Beskrivelse:
Pris: 3100 kr
Dage: 7
Ture: Cykeltur
Slet

Standard ture, gør at du kan oprette ture for rejsemålene, og slette dem. Du kan skrive pris, og antal dage i turene, og de vil blive inkluderet i Opret Rejseplan hvis du tilføjer turene.

TANA.Web

Home

Counter

Weather

Modify Email

Send Rejseplan

Opret Rejseplan

Rejser

Standard ture

login

Opret Standardtur

Titel Beskrivelse

Opret

Eksisterende ture:

- Guidet byvandring – 250 kr. – 1 dage
Slet
- Cykeltur – 100 kr. – 1 dage
Slet
- Svømmetur i søen – 200 kr. – 1 dage
Slet

Her har vi login siden, hvor man kan enten logge ind som en bruger (rejseagent), eller som en administrator.

Tanzania Eksperten

26-05-2025

EN

DA

Home

Kunde Oversigt

Modify Email

Send Rejseplan

Opret Rejseplan

Rejser

Standard ture

login

Template Builder

View All Templates

Login

Email

admin

Password

Login

Her kan man sende rejseplaner som PDF-filer til kunderne.

Tanzania Eksperten

26-05-2025

EN

DA

Home

Kunde Oversigt

Modify Email

Send Rejseplan

Opret Rejseplan

Rejser

Standard ture

login

Template Builder

View All Templates

Send rejseplan

Navn:

E-mail:

Rejseplan destination:

Afrejse:

Hjemrejse:

Flyseelskab:

Opret PDF

Skabalon oprettelse siden. Man kan oprette og tilpasse en skabelon til fremtidig brug.

Tanzania Eksperten

27-05-2025

EN

DA

Hjem

Kunde Panel

Redigér Emailadresse

Send rejseplan

Opret rejseplan

Rejser

Standard ture

Log ind

Opret skabelon

Skabeloner

Admin Panel

Template Builder

New Template (A4)

Add new section:

☐ Header ☐ Image ☐ Summary ☐ Day Sections

Customize Template:


Primary Color

Header Background Color

Footer Background Color

Font Family


Arial

 Tanzania Eksperten

Afhængig af oplevelsen for livet

Template Name:

Enter template name...

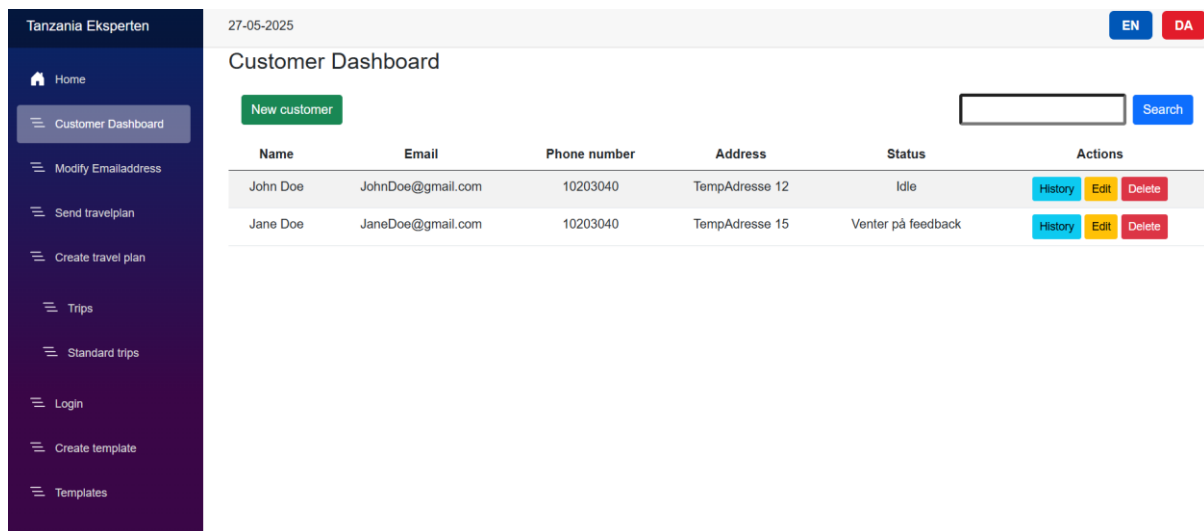
 Tanzania Eksperten Aps

CVR Nr. 41238425

Mobile: +45 42 73 10 45

Email: kontakt@tanzania-eksperten.dk

Kundeoversigt siden. Her kan man oprette nye kunder, se deres historik, eller lave opdatere deres information.



Diskussion

Da vi havde begrænset tid til at oprette vores system, endte vi med at ikke implementere alle de funktioner vi ville. Herunder er der funktioner som Produkt Owneren anmod. Vi sørgede for at fokusere på kernefunktionerne, så selv om vi ikke nåede alle de anmodet funktioner, har vi stadig skabt et system der vil være brugbart for Product-Owneren. De manglende funktioner er:

- Login for forskellige rejsebureauer – Det er kun muligt at logge ind som enten en bruger eller en admin.
- Kalendersystem – Kunden kan ikke direkte vælge tilgængeliges dage. De bliver nødt til i stedet at lægge en kommentar om hvilke dage de vil have rejsen foregår.
- Geografiske rutevisning.
- Oprettelse af betalingslink og besked når faktura er forfaldet - Agenten skal selv oprette en faktura og sende den manuelt gennem e-mail.

På grund af de manglende funktioner kan det nuværende system ses som bare en prototype der skal iterere på yderlige.

Der er selvfølgelig også mere funktionalitet der kunne implementeres for at skabe et bedre system. F.eks., siden diskussionen mellem rejseagenten og klienten er så vigtigt for produktejeren, kunne der tilføjes et mere direkte chatsystem mellem de to.

En anden retning at udvikle systemet ville være at fokusere mere på det administrative elementer. Vi fandt dog de af mindre vigtighed for systemet, da produktejeren var sig selv en rejseagent, og derfor ville flere administrative værktøj ikke være af stor nyttig for ham. Det kunne så diskuteres hvor vigtig disse værktøjer virkelig er for systemet og om deres inklusion er nødvendigt, da de ikke skaber værdi for produktejeren.

Konklusion

Som et team i dette projekt udviklede vi en webapplikation til Tanzania Expert (TANA) ved hjælp af SCRUM som procesmodel og Clean Architecture som et arkitekturprincip, der understøtter virksomhedens kernefunktioner:

Rejseplanlægning, hvor brugeren nemt kan oprette en rejseplan ved at vælge sektioner og nemt tilføje detaljer.

Kundeadministration hvor brugeren kan opdatere, slette og oprette kunder med oplysninger som adresse og e-mail.

Sikkerhed og adgang: Applikationen understøtter kryptering af følsomme data for både klienter og brugere. Hver brugers rolle kan bestemmes for hvilken side vedkommende vil blive dirigeret til.

Brugeren kan også oprette PDF-skabeloner og nemt genbruge dem, med mulighed for at ændre farve og skrifttype og sende dem til klienten via e-mail.

Systemet er udviklet i .NET 8 med Blazor og SQL ved hjælp af EF-migreringer, testet med xUnit og designet til at kunne udvides i fremtiden for at tilføje mere funktionalitet eller til vedligeholdelse på grund af ansvarsadskillelse gennem grænseflader.

Selvom er ikke alle funktioner er blevet implementeret pga. tidsbegrænsninger, såsom tilføje en interaktiv kortvisning, der viser brugeren rejsekortet og turistattraktioner, eller tilføje betaling ved at sende et link til kunden via e-mail, er blevet færdiggjort på grund af tidsbegrænsninger. Men vi har tilføjet grundlæggende funktioner til applikationen, der giver virksomheden fuld kontrol over processen med at oprette rejseplaner og sende dem til kunder.

Perspektivering

For at videreudvikle systemet i fremtiden kan vi for eksempel tilføje nye funktioner såsom at integration med korttjenester som Google Maps

eller tilføje login og gendannelsesfunktioner via MitID, hvis brugeren glemmer adgangskoden.

Virksomheden kan også tilføje nye betalingsmetoder, såsom PayPal, på grund af den nemme betaling, eller kortbetaling.

Kunde-loginfunktion kan også tilføjes for kunder, så de kan se, booke og betale for en specifik rejseplan uden behov for manuel indgriben fra personalet.

Med fremkomsten af AI-teknologier som chatbot baseret kundesupport kan der implementeres en chatbot, der kan modtage og besvare kundehenvendelser i løbet af ferier eller uden for normal åbningstid.

På grund af den begrænsede tid i projektet har vi fokuseret på at implementere kernefunktionaliteten. De nævnte funktioner blev ikke realiseret, men applikationen er arkitektonisk forberedt til fremtidig udvidelse. Takket være systemets lagdelte arkitektur og ansvarlig kodeopdeling, der består af flere lag, som er nemme at ændre.

Litteraturliste

- Funktionelle og ikke-funktionelle krav, Systemudvikling

[Funktionelle VS Ikke-funktionelle krav \(med eksempler\) - Visure-løsninger](#)

- Clean Architecture

<https://matthewrenze.com/presentations/clean-architecture/>

<https://github.com/matthewrenze/clean-architecture-core>

- OpenAI's ChatGPT som støtteværktøj til ideudvikling, sproglig formulering, stavekontrol.

<https://chatgpt.com>

- Kvalitetssikring og teststrategi, metrikker

[Generisk teststrategi](#)

Bilag

