Netcompany

Procesrapport

Gårdbutik

Emil Ømark Jensen
11-04-2023

Titelblad

TECHCOLLEGE

Techcollege Aalborg,

Struervej 70,

9220 Aalborg

Elev:
Emil Ømark Jensen

Firma:
Netcompany

Projekt:
Gårdbutik styring system

Uddannelse:
Datateknikker med special i programmering

Projektperiode:
11/04/2023 – 16/05/2023

Afleveringsdato:
04/05/2023

Fremlæggelsesdato:
dd/mm/yyyy

Vejledere:
Navn på vejledere

Indholdsfortegnelse

Titelblad i
Indledning 1
Case beskrivelse 1
Problemformulering 1
Projektplanlægning 1
Estimeret tidsplan 1
Arbejdsfordeling 1
Metode- og teknologivalg 1
Applikation 1
Frontend 2
Backend 2
Server 2
Asp.net core 2
Entitet framework 3
Blazor 3
Docker 3
PostgreSQL 4
Realiseret tidsplan 4
Konklusion 4
Diskussion 5
Tilføjelser 5
Referencer 5
Bilag 6

Indledning

En gårdbutik er en butik, som typisk ligger på en gård, hvor man kan købe produkter, der er enten produceret på gården eller lokalt producerede varer.

I en gårdbutik vil man kunne finde et udvalg af varer, herunder både gårdenes egne produkter og andre indkøbte varer, som normalt er andre vare som er produceret lokalt i nærheden. Alle varer, som sælges i butikken, skal kunne registreres i systemet, og det kan nogle gange være nødvendigt at samle produkterne i en gavekurv eller sælge dem i kilopris i stedet for stykpris.

Case beskrivelse

Som en lille gårdbutik vil der være mange forskellige priser og lager, der skal holdes styr på. Det kan være, om der er noget, der snart udløber, eller man skal kunne lave en faktura for en gavekurv med en masse forskellige ting i eller en bestilling af en samling af ting, hvor der skal findes ud af, hvad det koster for butikken at lave kurven, og hvad en god pris vil være for kurven. Derefter vil der også skulle holdes styr på, hvor meget salg og indtjening, man har haft.

Problemformulering

Problemet for gårdbutikken er at holde styr på, hvad der er på lager/i butikken, og hvornår noget udløber, samt at kunne oprette faktura for større bestillinger og lave statistik over hvad der er blevet solgt over det sidste stykke tid.

Projektplanlægning

Estimeret tidsplan

Den estimerede tidsplan er meget fleksibel, da der forventes at blive skiftet meget imellem forskellige opgaver. Der vil sandsynligt ikke blive afsat en dag til at arbejde på backend eller en anden del af projektet, samtidig vil der nok også blive skrevet lidt på produktrapporten og processrapporten.

På grund af denne tilgang vil det være udfordrende at opdele arbejdet i individuelle dage for hver enkelt opgave. Tidsplanen vil i stedet blive brugt som en deadline for, hvornår det forventes, at alt er 100% færdigt. (Bilag)

Arbejdsfordeling

I afsnittet "estimerede tidsplan" blev det beskrevet, at arbejdsfordelingen er meget fleksibel. Dette skyldes, at der sjældent vil være en dag, hvor der kun arbejdes på en opgave. I stedet vil man vælge at fokusere på den opgave, der mangler mest i forhold til hvornår deadline, der er blevet fastsat i tidsplanen. (Bilag)

Metode- og teknologivalg

Applikation

Denne applikation er en klient/server-webløsning, hvor frontenden er bygget med Blazor og backenden er bygget med ASP.NET Core. Formålet med applikationen er at håndtere og analysere data fra en database ved hjælp af en single-page-applikation (SPA). Brugerne vil kunne interagere med applikationen gennem et

responsivt og brugervenligt hjemmeside udviklet med Blazor. Backenden vil fungere som et API, der giver adgang til databasen og muliggør oprettelse, læsning, opdatering og sletning af data. Samt generering af dokumenter ud fra en templates lavet a brugen med word feild codes.

Frontend

frontenden er skrevet i Blazor, betyder det, at webapplikationen er udviklet ved hjælp af Blazor's Razor-komponenter og C#-kode, og at det kan køre direkte i browseren uden behov for ekstra JavaScript-biblioteker eller -frameworks.

Blazor er blevet valgt som grundlag for at skabe en Single Page Application (SPA), hovedsageligt på grund af dets brugervenlige sprog. Dette gør det muligt at holde det meste af koden inden for ét programmeringssprog, nemlig C#. Ved at benytte Blazor kan man dermed udvikle en mere strømlinet og effektiv applikation, som både er nem at arbejde med og vedligeholde.

Backend

Backenden er skrevet i ASP.NET Core, som er et open-sourceweb-applikations-framework, der er designet til at udvikle moderne, cloud-baserede applikationer på tværs af platforme. Det giver en række funktioner, der gør det nemt at opbygge en skalerbare og sikker web-applikation, herunder indbygget støtte til dependency injection og en fleksibel middleware pipeline.

ASP.NET Core er blevet valgt som backend-teknologi for at kunne oprette et API, der holder sig til det samme programmeringssprog, C#. Dette gør det også nemt at modularisere og udvide funktionaliteten ved hjælp af NuGet-pakker. For API'et er der anvendt et Model-View-Controller (MVC) designmønster, som sikrer en klar adskillelse af ansvar og gør det nemt at vedligeholde og skalere applikationen. På backend-siden er Entity Framework Core også valgt som en Object-Relational Mapping (ORM) løsning, hvilket muliggør nem styring af databasen ved hjælp af C#. Det forenkler integrationen mellem kode og database, og sikrer en mere ensartet og effektiv udviklingsproces.

Server

Applikationen vil blive hostet gennem Docker på en Linux-server, hvilket sikrer en ensartet og problemfri drift af både ASP.NET Core-webserveren og PostgreSQL-databaseserveren. Ved at anvende Docker-containere opnås en højere grad af fleksibilitet og skalérbarhed, samtidig med at man undgår eventuelle kompatibilitetsproblemer mellem forskellige miljøer. Denne opsætning er valgt for at sikre en effektiv, pålidelig og vedligeholdelsesvenlig infrastruktur for webapplikationen og databasen.

Asp.net core

ASP.NET Core er et åbent, højtydende og tværplatforms framework designet til at bygge moderne webapplikationer, der er optimeret til skyen og kan forbindes til internettet. Det fungerer på Windows, macOS og Linux og giver udviklere mulighed for at bruge deres foretrukne værktøjer.

Dette fleksible framework tilbyder en samlet tilgang til at skabe både webbrugergrænseflader og web-API'er. Det er udviklet med fokus på testbarhed og gør det nemt at kode sidefokuserede scenarier med Razor Pages. Desuden kan Blazor bruges til at skrive C#-kode i browseren sammen med JavaScript, hvilket giver mulighed for at dele logik mellem server og klient.

ASP.NET Core er open-source og fokuserer på samarbejdet med udviklerfællesskabet. Det understøtter integration af moderne klient-side frameworks og udviklingsprocesser samt hosting af RPC-tjenester med gRPC. Det har et miljøbaseret konfigurationssystem, der er klar til brug i skyen, og inkluderer indbygget afhængighedsinjektion og en modulær HTTP-anmodningspipeline, der er både letvægts og højtydende.

Dette framework kan hostes på forskellige servere, inklusive Kestrel, IIS, HTTP.sys, Nginx, Apache og Docker, og understøtter side-om-side versionering, hvilket forenkler moderne webudvikling.

Entitet framework

Entity Framework Core (EF Core) er en letvægts, ekstensibel og open-source ORM (Object Relational Mapper) fra Microsoft, der er brugt i denne applikation for at forenkle dataadgang og manipulation. EF Core gør det muligt at arbejde med databaser ved hjælp af .NET objekter og

LINQ-forespørgsler i stedet for at skrive SQL-kode direkte. Det hjælper med at reducere kompleksiteten af databasetilslutning, samtidig med at det sikrer en mere vedligeholdelsesvenlig og testbar kodebase. EF Core understøtter en række forskellige databaser, hvilket gør det til et fleksibelt og alsidigt valg for udvikling af moderne webapplikationer.

Blazor

Blazor er et frontend webudviklingsværktøj, der gør det muligt at bygge interaktive webapplikationer ved hjælp af C#-programmeringssprog og .NET-platformen. Det er en open source-teknologi, der giver udviklere mulighed for at skrive kode, der kører direkte i browseren, og som kan kommunikere med en backend API.

Blazor giver mulighed for at udvikle single-page applikationer (SPA) ved hjælp af Razor-komponenter, der kombinerer HTML og C#-kode på en effektiv måde. Det er også muligt at bruge Blazor til at opbygge traditionelle flersidede webapplikationer.

Docker

Docker-containere er standardiserede enheder af software, der pakker kode og alle dens afhængigheder sammen, så en applikation kører hurtigt og pålideligt på tværs af forskellige computer miljøer. En Docker containerbillede er en letvægts, selvstændig, eksekverbar pakke af software, der indeholder alt, hvad der er nødvendigt for at køre en applikation: kode, runtime, systemværktøjer, systembiblioteker og indstillinger.

Docker er blevet valgt som en central del af infrastrukturen for at kunne opnå en nem og problemfri implementering af serveropsætningen i forbindelse med projektet. Ved at bruge Docker-containere kan

man garantere en ensartet og pålidelig drift af applikationen på tværs af forskellige miljøer, hvilket gør det lettere at skalere og vedligeholde systemet. Dette valg sikrer en hurtig og effektiv implementeringsproces, der minimerer både udviklings- og driftsomkostningerne.

PostgreSQL

PostgreSQL er et kraftfuldt, open source objekt-relationelt databasesystem, der bruger og udvider SQL-sproget kombineret med mange funktioner, der sikkert lagrer og skalerer de mest komplekse datamængder. PostgreSQL har rødder tilbage til 1986 som en del af postgre-projektet ved University of California, Berkeley, og har mere end 35 års aktiv udvikling.

PostgreSQL har opnået et stærkt ry for sin pålidelige arkitektur, dataintegritet, robuste funktionssæt, udvidelsesmuligheder og det engagerede open source-fællesskab, der står bag softwaren. PostgreSQL kører på alle større operativsystemer og har været ACID-kompatibel siden 2001. Det inkluderer kraftige tilføjelser, såsom den populære PostGIS geospatiale databaseudvider.

PostgreSQL er det open source relationelle databasevalg for mange mennesker og organisationer takket være dets mange funktioner, der hjælper udviklere med at bygge applikationer, administratorer med at beskytte dataintegritet og opbygge fejltolerante miljøer, samt hjælpe dig med at håndtere dine data uanset størrelsen på datasættet. Udover at være gratis og open source er PostgreSQL meget udvidelsesdygtig, så du kan definere dine egne datatyper, opbygge brugerdefinerede funktioner og endda skrive kode fra forskellige programmeringssprog uden at rekompilere din database.

PostgreSQL er valgt som databaseløsning for at kunne tilbyde en gratis og omkostningseffektiv løsning for brugeren uden at gå på kompromis med ydeevne og pålidelighed. PostgreSQL er et open source objekt-relationel databasesystem, der fungerer godt sammen med ASP.NET Core og tilbyder en stærk og fleksibel databaseløsning, der er i stand til at understøtte en lang række applikationer og behov. Dette valg sikrer en solid og omkostningseffektiv infrastruktur, der kan tilpasses efter projektets krav og vækst.

Realiseret tidsplan

Den realiserede tidsplan har fulgt den estimerede tidsplan ret nøje. Dog er der enkelte forsinkelser i forhold til programmet og dets test, som ligger en smule bagud i forhold til det forventede.

Se (Figur1 Tidsplan)

Konklusion

I denne rapport er der blevet præsenteret en løsning til at hjælpe små gårdbutikker med at holde styr på deres lager og produkter, administrere fakturering af større bestillinger samt analysere salgsdata. Applikationen er bygget som en klient/server-webløsning, hvor frontenden er udviklet i Blazor og backenden i ASP.NET Core.
Ud fra case beskrivelsen og problemformuleringen har vi udviklet en applikation, der adresserer de udfordringer, som gårdbutikker oplever i forbindelse med styring af lager, udløbsdatoer, fakturering og

salgsstatistik. Applikationen giver butikken mulighed for nemt at holde styr på alle produkter, både egne og indkøbte varer, samt at lave fakturaer for gavekurve og større bestillinger.

Under processen har jeg opnået nye færdigheder og forbedret eksisterende kompetencer inden for forskellige teknologier. Blandt de nye teknologier, jeg lærte at arbejde med, er Blazor og PostgreSQL, som jeg ikke har anvendt tidligere.
Desuden har jeg også genopfrisket min viden om andre teknologier, som jeg tidligere har arbejdet med, men ikke i lang tid. Disse inkluderer ASP.NET Core, Docker og Entity Framework. Denne kombination af at lære nye teknologier og genopfriske eksisterende viden har været både udfordrende og givende, hvilket har bidraget til min personlige og faglige udvikling.

Diskussion

Tilføjelser

Der vil også blive udviklet en funktion i applikationen, som giver kunderne mulighed for nemt at scanne varer og tilføje dem til en samlet pris. Når kunden har scannet alle varer og er klar til at betale, vil systemet automatisk åbne MobilePay med det samlede beløb, hvilket gør betalingsprocessen både hurtig og bekvem. Denne funktion vil forbedre kundeoplevelsen og effektiviteten ved indkøb, samtidig med at det gør betalingsprocessen mere strømlinet for kunder.

Referencer

ASP.NET Core blazor: https://learn.microsoft.com/da-dk/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0

ASP.NET Core blazor historie: https://en.wikipedia.org/wiki/Blazor

ASP.NET Core https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0

PostgreSQL https://www.postgresql.org/about/licence/

PostgreSQL https://www.postgresql.org/about/

Docker https://www.docker.com/resources/what-container/

Bilag

Figur1 Estimeret og realiserede- tidsplan

Figur2 Dagbog

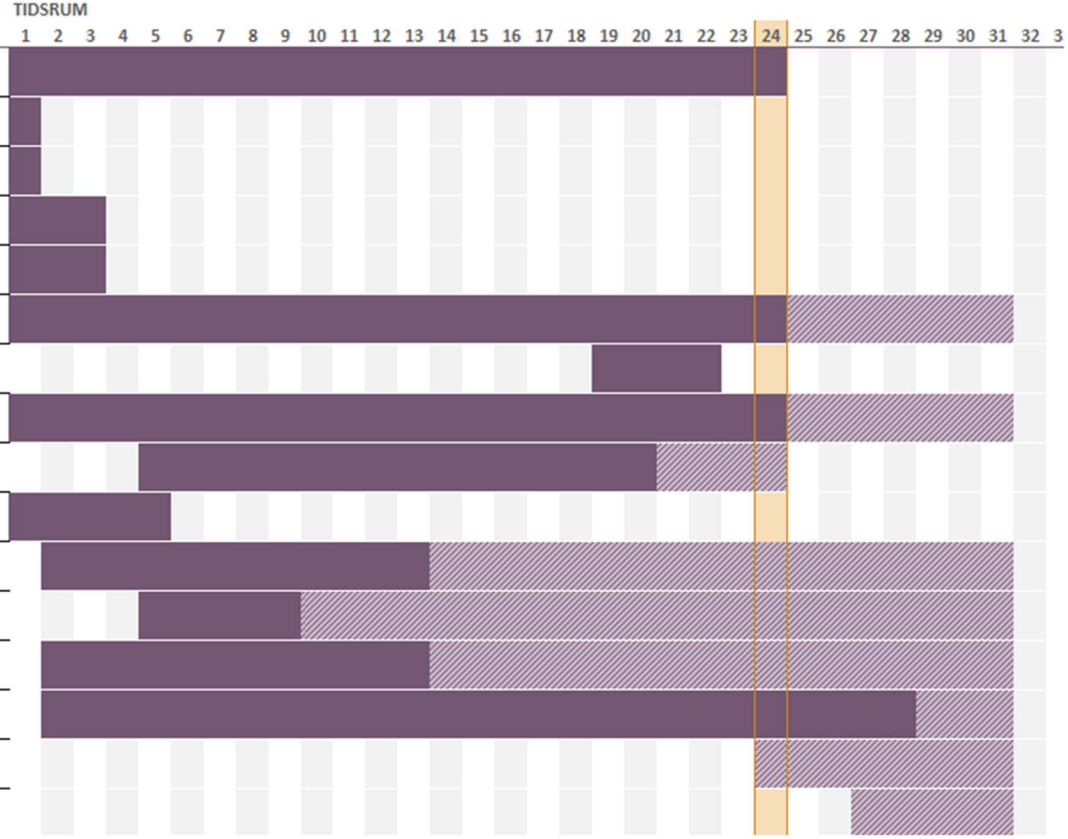
Projektplanlægger

Fremhæv et tidsrum til højre. Derpå følger en forklaring af diagrammet.

Fremhævning af tidsrum: 24

Varighed for plan | Faktisk startdato | % fuldført | Faktisk (ud over plan) | % fuldført (ud over plan)

AKTIVITET	STARTDATO FOR PLAN	VARIGHED FOR PLAN	FAKTISK STARTDATO	FAKTISK VARIGHED	PROCENTDEL FULDFØRT
Processrapport	1	24	1	24	100%
Case beksrivelse	1	1	1	1	100%
Problem fomulering	1	1	1	1	100%
Projektplanlægning	1	3	1	3	100%
Metoder og teknologi	1	3	1	3	100%
Realiseret tidsplan	1	31	1	31	80%
Konklusion	19	4	19	4	100%
Dagbog	1	31	1	31	80%
Produktrapport	5	20	5	20	80%
Kravspecification	1	5	1	5	100%
Produkt	2	30	2	30	40%
Frontend	5	27	5	27	20%
Backend	2	30	2	30	40%
Server	2	30	2	30	90%
Test	24	8	24	8	0%
Selv valgt emne	27	5	27	5	0%



Figur I Biag A: Estimeret Tidsplan

Dagbog/Logbog

Dag 1:

Skrev problemformulering og casebeskrivelse.

Dag 2:

Oprettede git repository og begyndte på tidsplanen.

Dag 3:

Skrev indledningen og færdiggjorde tidsplanen.

Dag 4:

Oprettede projektet og stødte på problemer med Docker.

Dag 5:

Brugte meget tid på Docker og arbejdede lidt på projekt opsætning, samt begyndte på vejledningen.

Dag 6:

Arbejdede ikke på noget projektrelateret.

Dag 7:

Udviklede front-end og skrev dagbog for ugen.

Dag 8:

Skrev procesrapport og havde problemer med Dockerfile.

Dag 9:

Skrev procesrapport og besluttede ikke at starte på tests, da projektet er længere bagud end forventet.

Dag 10:

Skrev procesrapport.

Dag 11:

Arbejdede på begge rapporter og begyndte på front-end til programmet.

Dag 12:

Slappede af og arbejdede ikke på svendeprøven.

Dag 13:

Skrev på rapporten.

Dag 14:

Skrev på rapporten.

Dag 15:

Arbejdede både på rapport og program.

Dag 16:

Arbejdede både på rapport og program.

Dag 17:

Skrev på rapporten.

Dag 18:

Skrev på rapporten.

Dag 19:

Arbejdede ikke på svendeprøven.

Dag 20:

Arbejdede ikke på svendeprøven.

Dag 21:

Arbejdede på programmet.

Dag 22:

Skrev på rapporten.

Dag 23:

Skrev på rapporten.

Dag 24:

Afleverede rapporten.