**Sluttrapport Coach Connect AS**

Coach Connect AS er et Rest-API for trenere og foreldre, og det gjør det enklere for begge parter å se treninger, kamper og statistikk for spillerne.

Hovedmålet er å få opp en spillerstatistikk over treninger og kamper for trenere. Dette skulle brukes videre på slutten av året for å kåre årets innsats, samt slik at trenere kan sjekke opp hvor mange kamper og treninger en spiller har deltatt på for å bruke dette videre til å sette opp lag for kamper og treningsgrupper.

Prosjektets formål er å effektivisere organiseringen av treninger, kamper og spillerstatistikk. Gjennom en kort introduksjon til prosjektet, problemstillingen, målene og metodene som er brukt gir rapporten en helhetlig oversikt over prosjektet.

Coach Connect AS-prosjektet tar sikte på å løse utfordringer knyttet til organiseringen av treningsøkter, kamper og spillerstatistikk. Med en stadig økende interesse for sport og aktiviteter, er behovet for effektive verktøy som kan håndtere denne organiseringen stort. Målgruppen for prosjektet inkluderer trenere, foreldre og spillere, og prosjektet søker å tilby løsninger som kan bidra til å forbedre kommunikasjonen og samarbeidet mellom disse gruppene.

Metodikken for å utvikle Coach Connect AS omfatter en grundig analyse av behovene til målgruppen, som i utgangspunktet var Quyen som er trener for Sandefjord Ballklubb G 2015.  Gjennom samarbeidet oss imellom har teamet bak prosjektet kontinuerlig tilpasset og forbedret løsningen. Denne sluttrapporten inkluderer også en gjennomgang av samarbeidsprosessen internt i teamet, med blant annet den fordelingen vi har hatt av arbeidsoppgaver.

Resultatene av Coach Connect AS-prosjektet inkluderer et funksjonelt REST API som muliggjør opprettelse av treningsøkter, kamper, påmelding av spillere og tilgang til spillerstatistikk. Testing og evaluering av programvaren har vist lovende resultater med hensyn til effektivitet og brukervennlighet, men med rom for forbedringer.

Vi testet endepunktene ofte underveis og laget tester for hver controller for å sørge for at apiet kjører som den skal.

Diskusjonen tar for seg betydningen av resultatene i forhold til prosjektets mål og formål. Videre drøftes potensielle forbedringer av løsningen og håndtering av eventuelle begrensninger, og inkludert muligheten for å skalere og utvide funksjonaliteten til Coach Connect.

Ting vi kunne gjort annerledes:

1. Vi erkjenner at vi gjorde en betydelig feil i den innledende planleggingsfasen angående databasesystemet og API-designet, med tanke på hvor viktig det er at det skal være skalerbart. Det skapte en del problemer underveis som vi måtte kode rundt med unødvendig kode og workarounds. Det ble for mye refaktorering som skulle til hvis vi hadde endra på hele database strukturen og apiet da vi var godt i gang med koding allerede. Dette skyldes vår manglende erfaring på det tidspunktet.

Til tross for dette fortsatte vi fremover, og ser nå på denne opplevelsen som en verdifull læringsmulighet. Det har gitt oss en dypere forståelse av viktigheten av å legge riktig planlegging til grunn fra begynnelsen av, og oppfordrer oss til å være mer forutseende i fremtidige prosjekter.

Vi kan nevne f eks følgende designfeil / forbedringer vi nå ville gjort:

* Man kan ikke være f eks både **User** og **Coach** uten å bryte normalisering i databasen - løsning -> vi skulle kun hatt en felles tabell for alle brukere og heller hatt roller her.
* Et team kan bare ha en **coach** -> løsning **TeamCoachRegister** (join-table). Må ha **RegisterCoachToTeam** funksjon da også (man må også dette godkjent av hovedtrener)
* En Player kan bare være på ett lag -> løsning **TeamPlayers** (join-table)
* En Player kan bare ha en User (forelder el likn) linket til seg -> løsning **UserPlayers** (join-table)
* **Jwt\_User\_Roles ->** tabell skulle hatt kobling med Users Table, men siden vi har **User** og **Coach** separat (som nevnt feil) så ble det workaround for å komme videre.
* **User** og **Coach** har ekstremt mye “dobbel” kode (egentlig alt), som er helt unødvendig, og ville vært løst dersom vi hadde felles **UsersTable** med roller. Dette har skapt mye hodebry som har tatt lang tid å løse underveis.
* **Returning of null** -> Vi returnerer nå null overalt dersom en request ikke går gjennom, nå kun med log for forklaring på hvorfor BadRequest trigges. Vi burde hatt custom exceptions el liknende for å formidle hva gjorde dette.
* **Blazor ->** Vår begrensede erfaring gjorde det utfordrende å forstå sammenkoblingen mellom **Blazor** og **API-et**. Dette førte til betydelige forsinkelser da vi ikke klarte å finne en løsning innenfor den tilgjengelige tidsrammen.

Ting vi ikke rakk å gjøre / burde vært med:

**1. Caching**

**2. HateOAS**

**3. NginX/YML**

**4. Blazor Frontend**

**5. Mer detaljert statistikk over oppmøte for spillere, med graf el diagram, over feks en tidsperiode.**

**6. Forskjellige type testere**

**7. Email/sms/push varsel for kamper og treninger til foreldrene**

**Hva er det viktigste vi har lært av å jobbe i gruppe og det å starte et prosjekt fra bunn:**

Hvor viktig det er å planlegge grundig fra start, og ta seg god tid til denne prosessen. Vi fant ut at det vi planla på starten av prosjektet ikke var godt nok.

Å jobbe i gruppe har også gitt oss mye lærdom når det gjelder det å samarbeide som et team på et felles prosjekt. Det har vært en veldig lærerik prosess for oss alle.

Vi ble godt kjent med Git og Github under prosjektet, vi fikk en del erfaringer med bla merge konflikter og andre Git issues man kan støte på underveis, og hvordan dette kunne løses.

Vi brukte mye tid på å lære og implementere ting som JWT Token og Guid, noe som var nytt for oss. Spesielt var overgangen til å bruke Guid i stedet for bare INT som ble ganske annerledes og påvirket mange deler av prosjektet.

Vi har innsett hvor lang tid ting egentlig tar i et slikt prosjekt, hvertfall for oss som studenter, hvordan feks en liten feil/fiks kan ta timesvis å finne og å fikse, eller å finne en løsning for logikk som skal skrives. Hvordan skal man få tak i den data man trenger f eks.

**Programmer / hjelpemidler som vi har brukt underveis:**

* Teams
* Trello
* Visual Studio / C# .NET Core / Entity Framework
* Visual Studio Code
* Swagger / PostMan
* MySQL / MySQL Workbench
* Github - <https://github.com/Salar977/CoachConnect>
* Git
* Google Docs
* Powershell / Git bash
* Docker / Docker Hub
* Draw.io

Hvem har gjort hva:

**Ketil og Quyen** (parprogrammering og individuelt):

* Strongly typed GUIDS inkludert OnModelcreating(modelbuilder) setup
* Serilog
* Login system/ JWT Autentisering / Middleware Extraction token etc
* Testdata script for databasen inkl Passwords.txt - lagt inn all data i scriptet
* UsersController -> logikk, mappere, dtos, validering osv
* CoachesController ->  logikk, mappere, dtos, validering osv
* GamesController -> logikk, mappere, dtos, validering osv
* GameattendancesController -> logikk, mappere, dtos, validering osv
* Integrasjonstester for User, Coach, Games, GameAttendances
* Oppsett av Docker container for database med alle scripts som trengs etc for testing (folder Docker/MySQL)
* HTTP User, Coach, Game, Gameattendance
* Dokumentasjon for Login, User, Coach, Game, GameAttendances
* Global Exception handling

**Kristian**:

* Player (Controller, Service, Repository, FluentValidations)
* Team (Controller, Service, Repository, FluentValidations)
* Player Integrasjonstester
* Team Integrasjonstester
* ReadMe.md for Github

**Salar:**

* N-tier Arkitekturen (Fil og mappestruktur)
* Fordele Dependency injection for et finere Program.cs
* Practice (Controller, Service, Repository, Dtos, Mappers)
* PracticeAttendance (Controller, Service, Repository, Dtos, Mappers)
* Entity FrameworkCore + Migrering gjennom forskjellige prosjekter
* HTTP practice and practice attendance
* Rate limiting
* Miljøvariabler for database brukernavn og passord
* Integrasjon Tester for practice and practice attendance
* FluentValidation for Practice og PracticeAttendance

Vi har reflektert over prosjektets utfordringer og suksesser, og vi er stolte av hva vi har oppnådd til tross for noen feil underveis. Denne erfaringen har vært uvurderlig for vår personlige og faglige utvikling, og vi er takknemlige for muligheten til å lære og vokse gjennom dette prosjektet.

**UML-Diagrammet for Databasen:**

