SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

FOOTBALLXG WEB-SOVELLUS

TEKIJÄ/T: Arttu Reijonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Arttu Reijonen	
Työn nimi FootballxG Web-sovellus	
Päiväys 07.01.2020	Sivumäärä/Liitteet 33
Ohjaaja(t) Mikko Pääkkönen, Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Yksityishenkilö	

Tiivistelmä

Projektin tavoitteena oli kehittää moderneja web-kehitysmenetelmiä käyttäen asiakkaan tarpeisiin soveltuva web-sovellus. Sovelluksella pystytään laskemaan, tallentamaan, poistamaan, muokkaamaan ja katselmoimaan jalkapallossa käytettävää xG-dataa. xG-datalla tarkoitetaan maaliodottamaa, jolla ilmaistaan mahdollisuutta tehdä maali kyseisestä laukaisupaikasta. xG-datan laskentaan käytettiin apuna matemaattista kaavaa, jonka avulla saatiin laskettua tarkkoja xG-lukuja.

Teoriaosuudessa käytiin läpi projektissa käytetyt teknologiat, joita ovat Angular8, ASP.NET Core, Github sekä Microsoftin tuotteet Visual Studio, Azure ja VScode.

Kehitysosuudessa selostettiin sovelluksen rakennetta, toiminallisuuksia ja miten sovellus toimii käytännössä. Luvussa esiteltiin selkä kokonaiskuva sovelluksen käytettävyydestä ja sen kehitysprosessista.

Viimeisessä luvussa tehtiin yhteenveto laadittuihin tavoitteisiin pääsemisestä, aikataulusta, mitä opittiin ja mitä ongelma tilanteita kohdattiin projektin toteutuksessa. Yhteenvedossa esitellään myös jatkokehitysideoita, joita kirjattiin projektin edetessä.

Avainsanat

SAVONIA UNIVERSITY OF APPLIED SCIENCES THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Arttu Reijonen			
Title of Thesis FootballxG Web Application			
Date	7 January 2020	Pages/Appendices	33
Supervisor(s)			
Client Organisation /Partners A private person			

Abstract

The subject of this thesis was to create a web application by using modern software development tools and technologies. The goal was to create a web application that can be used to calculate, save, update, delete, modify and to view calculated xG data which is used to track players' performance in football. xG data stands for expected goals and it is used to express the possibility to make a goal at the current shot location with the specific parameters and formula. The web application can be used to collect xG data from matches, practices and players.

FootballxG was designed by using modern web development technologies. These technologies are Angular8, ASP.NET Core 2.2, Github and various Microsoft's products including Visual Studio, Visual Studio Code and Microsoft Azure. The web application was designed based on research on various coding tutorials and online documentation sources. Technologies and tools were chosen to suit the developed application.

The end result of this project is a modern web application which is hosted on a Microsoft Azure server.

Keywords

SISÄLLYSLUETTELO

1 JOHDANTO

Opinnäytetyön tavoitteena on luoda web-sovellus, jonka avulla käyttäjät voivat laskelmoida maaliodottamien todennäköisyyksiä, sekä tallentaa näitä tietoja tietokantaan. Maaliodottama eli xG on jalkapallossa käytettävä termi, joka tarkoittaa todennäköisyyttä millä laukaisu menee vastustajan maaliin. xG laskennassa käytetään laskukaavaa, johon sovelletaan laukaisulle yksilöllisiä parametrejä. Maaliodottamien laskennassa otetaan huomioon erilaisia muuttujia, kuten laukaisupaikka, ruumiinosa ja pelitilanne.

Datan keräämisestä on tullut tärkeä osa urheiluja ja joukkueet, joilla on vaadittavat resurssit datan keräämiseen, saavat tärkeän etulyöntiaseman pelaajien kehittymisen ja otteluiden lopputuloksen suhteen. Maaliodottamien laskemisen ja tallettamisen suhteen joukkueet hyötyvät vastustajien datan tarkastelemisesta, jotta he voivat tehdä tarvittavia muutoksia omaan pelisuunnitelmaansa ja yksittäiset pelaajat voivat kehittää omia taitojaan tarkastelemalla heistä kerättyä dataa.

Työn tilaaja on yksityishenkilö, jonka kanssa sovellus suunniteltiin ja laadittiin tekninen dokumentaatio, jossa kuvattiin vaadittavat ominaisuudet sovellusta varten. Kehitysympäristönä projektissa käytettiin Microsoft Visual Studiota ja palvelinpuolena toimii ASP.NET Core 2.2 versio. Sovelluksen käyttöliittymä toteutettiin käyttäen Angular8 ohjelmistokehystä.

1.1 Termit ja lyhenteet

Frontend = Frontend eli selainpuolella tarkoitetaan kaikkea sitä koodia, joka ajetaan verkkoselaimessa. Frontendiä ovat esimerkiksi sivun rakenne (html), ulkoasu (css) ja selaimessa tapahtuvat toiminnallisuudet (javascript). Tässä projektissa käytetään myös Typescriptiä, joka on eräänlainen javascript kirjasto.

Backend = Backend tarkoittaa koodia, joka ajetaan sivuston palvelimella. Täällä tapahtuvat esimerkiksi sellaiset asiat kuin lomakkeiden käsittelyt, kirjautuminen ja salasanojen tarkistaminen, järjestelmäintegraatiot ja tietokantojen käsittely.

xG (Expected Goals) = Odotettu maalimäärä, jota käytetään jalkapallossa maalitodennäköisyyden laskemiseen. xG:llä ilmoitetaan lukua millä todennäköisyydellä saadaan tehtyä maali kyseisestä vetopaikasta, kyseisillä parametreillä.

HTTP (*Hypertext Transfer Protocol*) = http on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon. Protokolla perustuu siihen, että asiakasohjelma avaa TCP-yhteyden palvelimelle ja lähettää pyynnön. Palvelin vastaa lähettämällä sopivan vastauksen, tavallisimmin HTML-sivun tai binääridataa kuten kuvia, ohjelmia tai ääntä. (Oamk, tietoliikenneohjelmointi)

Sprintti (*Sprint*) = Sprint on enintään kuukauden pituinen tai sitä lyhyempi aikaraja, jonka sisällä tuotetaan “valmiin” määritelmän täyttävä, käyttökelpoinen ja potentiaalisesti julkaisukelpoinen tuoteversio. Sprinteillä on sama pituus koko kehityksen ajan. Uusi sprintti alkaa välittömästi edellisen päätyttyä.

CRUD (Create, Read, Update, Delete) = CRUD on tiedon tallentamisen neljä perustoimintoa.

- CREATE-menettelyt: Suorittaa INSERT-käsky uuden tietueen luomiseksi.
- READ-menettelyt: Lukee taulukkotietueet syöttöparametrin.
- UPDATE-menettelyt: Suorittaa UPDATE-käskyn taulukossa määritetyn ensisijaisen avaimen perusteella tietueelle WHERE-lauseen sisällä.
- POISTA menettelyt: Poistaa määritellyn rivin WHERE-lausekkeesta. (Heller Martin, 2017)

2 ODOTETUT TAVOITTEET JA NIIDEN LASKEMINEN

”Odotetut tavoitteet eli xG on jalkapallomittari, jonka avulla voidaan arvioida joukkueen ja pelaajien suorituskykyä. Matalan pistemäärän omaavissa peleissä, kuten jalkapallossa, ottelun lopputulos ei anna selkeää kuvaa pelaajien ja joukkueen suorituskyvystä. Siksi yhä useammat urheiluanalyysit kääntyvät edistyneisiin malleihin, kuten xG, joka on tilastollinen mittari luotujen ja hyväksyttyjen mahdollisuuksien laadusta.” (Hyppänen, 2017) xG-laskennan tarkoituksena on tilastoida ottelun tärkeimmät tapahtumat numeerisessa muodossa. Ideana on tilastoida sellaiset tapahtumat, joiden riittävä toisto johtaa ajan myötä maaleihin. Tällaisia ovat esimerkiksi laukaukset, puskut, kulmapotkut, vapaapotkut, rangaistuspotkut ja keskitykset maalinedustaan.

Opinnäytetyön tavoitteena oli luoda mahdollisimman tarkka menetelmä laukauksien laadun arviointiin ja tallettamiseen. Maaliodottamien laskentaan käytetään laskukaavaa, joka ottaa huomioon laukaisun paikan maaliin nähden, kulman, ruumiinosan, jolla laukaisu suoritettiin ja pelitilanteen Laskukaavassa käytetään useita muuttujia ja parametrejä, joiden avulla saadaan tarkka xG-luku.

Projektissa xG-dataa lasketaan kolmella eri alasivulla, jotka ovat ottelut, harjoitukset ja joukkueet. Tämä tarkoittaa sitä, että käyttäjät voivat kerätä xG-dataa otteluista, harjoituksista, sekä luomaan yksilöllistä pelaaja dataa joukkueille. Jokaisella alasivulla luodut xG-datat tallennetaan tietokantaan, josta niitä voidaan tarpeen vaatiessa katselmoida, muokata tai poistaa käyttäjän tarpeen mukaan. Alla kuva (Kuva 1) Hector Bellerin laukaisusta, jolla xG oli 0.33.

Kuva 1 Hector Bellerin laukaisu Leicesteriä vastaan. (Skysports, 2017)

3 TIETOKANTA

Tietokannan suunnittelu aloitettiin määrittelemällä sovelluksen käyttötarpeet. Aloituspalaverissa todettiin, että sovelluksessa pitää olla käyttäjien autentikointi, sekä mahdollisuus tallentaa maaliodottamien muodostamaa dataa tietokantaan. Sovellus käyttää yhtä tietokantaa, mutta se on jaettu kahteen osaan, jotka ovat käyttäjän autentikointi ja xG laskennassa käytetty ja tallennettu data. Näitä tietokantatauluja hallinnoidaan sovelluksen taustajärjestelmässä kahdella DbContext-luokalla, joilla voimme välittää tiedot oikeisiin tietotauluihin tietokannassa.

Tietokannan luontiin käytettiin Code first menetelmää, jossa tietokantataulut luotiin tekemällä niistä malliluokkia ja yhdistämällä ne DbContext-luokkaan. Tämän jälkeen tehtiin migraatio, jonka avulla lähetettiin DbContextin sisältä SQLexpress palvelimelle, joka loi migraation avulla tarvittavat taulut. Lokaalissa kehityksessä (kehitys, joka tapahtuu kehittäjän tietokoneella) käytettiin Microsoftin tarjoamaa serveriä SQLexpressiä. SQLexpressin avulla voidaan luoda palvelin, jonne tietokanta säilötään. SQLexpresiin otetaan yhteys SSMS avulla. Visual Studio projektista pystyttiin ottamaan yhteys tähän SQLexpress palvelimeen ja täten luomaan yhteys tietokannan sovelluksen välillä. Projektin julkaisussa käytettiin Microsoft Azuren tarjoamaa palvelinta. Sovelluksen julkaisun yhteydessä Azure loi palvelimen, johon pystyttiin ottamaan yhteys projektin connection stringin avulla ja täten luomaan tietokanta Azuren palvelimelle.

Tietokannan suunnittelussa käytettiin apuna dbdiagram.io sivustoa, jossa pystyttiin mallintamaan tietokannan taulujen relaatioita ja tekemään suunnitelmia ympäristössä, jossa virheillä ei ollut merkitystä. Kun tietokanta saatiin suunniteltua, siirryttiin tietokannan luontiin Visual studiossa.

3.1 Käyttäjien autentikointi JSON Web Tokenilla

Käyttäjien autentikointiin käytettiin avointa JSON Web Tokenia (JWT). ”JSON Web Token itsessään on merkkijono, joka koostuu kolmesta osasta: otsikko (header), sisältö/väitteet (payload/claims) ja kolmantena salaisesta osasta, ja muusta sisäl-

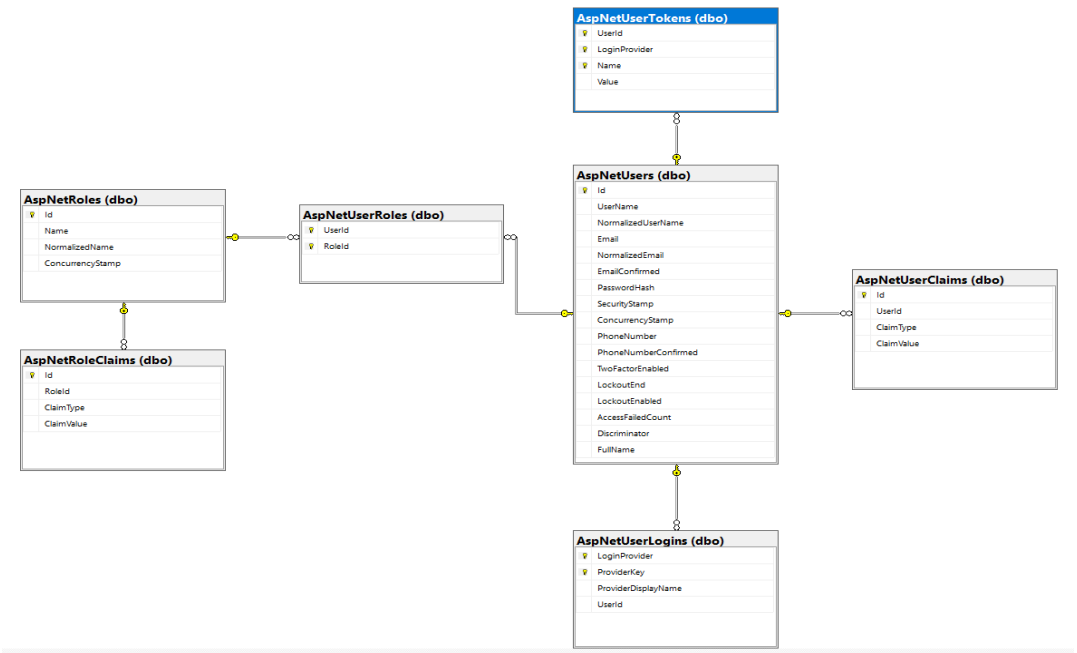
löstä (edellä mainitut) yhteensä muodostamasta tiivisteestä. Nämä tiedot erotetaan toisistaan pisteellä ja muutetaan base64-muotoon.” (Auth0 / JWT Documentation). Pyyntöä tehtäessä palvelin varmentaa salausavaimella, onko käyttöoikeustietue kunnollinen, eli voidaanko pyynnön katsoa olevan turvallinen. Alla kuvassa (Kuva 2) esimerkki muodostetusta JSON web tokenista.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Kuva 2. Esimerkki JWT tokenista (Auth0 / JWT Documentation)

3.2 Käyttäjien tietokantarakenne

Tietokannan ja sovelluksen rakentaminen aloitettiin ensin rakentamalla käyttäjien autentikointi, alla kuva käyttäjien tietokanta toteutuksesta, jonka toteuttamiseen käytettiin ASP.NET Identity pohjaa. (ASP.NET Docs, 2019)

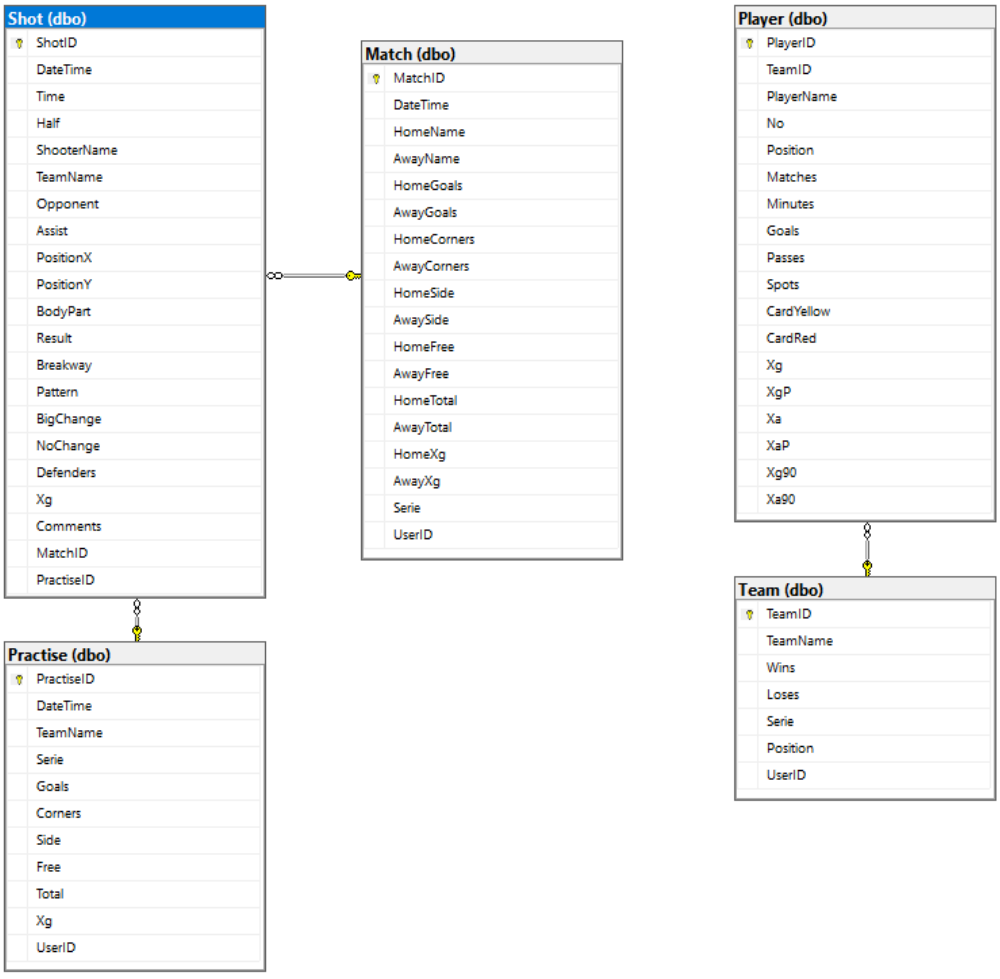


Kuva 3. Käyttäjien tietokantarakenne (Reijonen, 2019)

Kuvassa (Kuva 3) näytettävän tietokannan keskeisimpänä tauluna toimii AspNetUser taulu, johon tallennetaan nimensä mukaisesti käyttäjien keskeisimmät tiedot.

3.3 Maaliodottamien tietokantarakenne

Kun käyttäjien taulut saatiin rakennettua kantaan ja onnistuttiin toteuttamaan käyttäjien autentikointi sovelluksessa, aloitettiin maaliodottamien tallentaminen kantaan. Alla olevassa kuvassa (Kuva 4) kuvataan maaliodottamien tallennusta kantaan.



Kuva 4. xG-datan tietokantarakenne (Reijonen, 2019)

Tauluilla Practise ja Match on vierasavain relaatio tauluun Shot. Tällöin kummatkin taulut voivat käyttää hyväkseen Shot taulua, ja täten pystytään tallettamaan otte-

luille ja harjoituksille yksilöllisiä xG laskentoja. Team taululla on vierasavain relaatio Player tauluun. Tämä mahdollistaa sen, että joukkueelle voidaan lisätä yksilöllisiä pelaaja tietoja.

Kolmelle päätason taululle eli tauluille Practise, Match ja Team on lisätty kenttä ”UserID”, johon talletetaan POST kutsun yhteydessä kirjautuneen käyttäjän ID, joka mahdollistaa sen, että kaikki data mitä tallennetaan, on yksilöity vain kyseiselle käyttäjälle. UserID saadaan kätevästi käyttöön Claimsista. Käyttäjän tiedot tallennetaan Claimiin käyttäjän kirjautumisen yhteydessä. Alla kuva Http Get kutsusta Match controlleriin, jossa UserID haetaan claimistä ja otteluiden tietokantahaut tehdään UserID perusteella. Alla olevassa kuvassa (Kuva 5) nähdään miten UserID haetaa claimistä Matchcontrollerin Get funktiossa.

```
// GET: api/Match
[HttpGet]
0 references | Arttu Reijonen, 41 days ago | 1 author, 2 changes | 0 requests | 0 exceptions
public async Task<ActionResult<IEnumerable<Match>>> GetMatch()
{
    // return await _context.Match.ToListAsync();

    string userId = User.Claims.First(c => c.Type == "UserID").Value;

    var match = (from a in _context.Match
                 where a.UserID == userId
```

Kuva 5. MatchController Get kutsu (Reijonen, 2019)

4 TEKNIIKAT JA KIELET

Tässä luvussa kuvataan sovelluksessa käytettyjä tekniikoita. Sovelluksen toteutuksessa on käytetty yleisimpiä web-tekniikoita. Ulkoasu on toteutettu Bootstrapilla, jota on muokattu CSS-kielellä projektin vaatimusten mukaan. Sovelluksen FrontEnd on toteuttu Angular8 käyttäen ja BackEnd on toteutettu ASP.NET Core 2.2 pohjalle.

4.1 Angular 8

Angular on avoimen lähdekoodin Typescript-pohjainen ohjelmistokehys. Alkuperäinen AngularJS julkistettiin vuonna 2009, uusin versio Angular 8 julkistettiin toukokuussa 2019 ja Angular on nykyisin Googlen ylläpitämä avoimenlähdekoodin palvelu. (Angular8 Documentation)

4.1.1 Components

Angularin avulla voidaan luoda projektiin yksittäisiä komponentteja, joilla jokaisella on oma HTML-, CSS-, ja Typescript tiedostonsa, joka helpottaa sovelluksen käytettävyyttä ja ymmärtämistä.

```
5  @Component({
6    selector: 'app-home',
7    templateUrl: './home.component.html',
8    styleUrls: ['./home.component.css']
9  })
```

Kuva 6. Home-component.ts tiedoston selector, templateUrl ja styleUrls (Reijonen, 2019)

Yllä olevassa kuvassa (Kuva 6) Home componentin Typescript tiedoston @Component rakenne, jossa määritellään selector eli miten componenttia kutsutaan eri

tiedostoissa. TemplateUrl kertoo html tiedoston sijainnin ja StyleUrls kertoo tyylitiedostojen sijainnin. Selectoria voidaan kutsua parent tiedostossa <app-home></app-home> tagin avulla, jolloin componentti renderoityy sivustolle käyttäjälle näkyviin.

4.1.2 Directives

Angularissa on valmiita direktiivejä, jotka on rakennettu helpottamaan sovelluksen kehittämistä. Alla olevassa kuvassa (Kuva 7) on esimerkki *ngFor direktiivistä, jota voidaan käyttää apuna looppaamaan läpi teamList objekti.

```
<tr *ngFor="let team of teamList | filter:searchText">
  <td (click)="openForEdit(team.TeamID)" *ngIf="team.Serie == this.sortSerie || this.sortSerie == ''">{{team.TeamName}}</td>
  <td (click)="openForEdit(team.TeamID)" *ngIf="team.Serie == this.sortSerie || this.sortSerie == ''">{{team.Serie}}</td>
  <td (click)="openForEdit(team.TeamID)" *ngIf="team.Serie == this.sortSerie || this.sortSerie == ''">{{team.Position}}</td>
  <td (click)="openForEdit(team.TeamID)" *ngIf="team.Serie == this.sortSerie || this.sortSerie == ''">{{team.Wins}}</td>
  <td (click)="openForEdit(team.TeamID)" *ngIf="team.Serie == this.sortSerie || this.sortSerie == ''">{{team.Loses}}</td>
  <td  *ngIf="team.Serie == this.sortSerie || this.sortSerie == ''">
    <a class="btn text-danger" (click)="onOrderDelete(team.TeamID)"><i class="fa fa-trash fa-lg"></i></a>
  </td>
</tr>
```

Kuva 7. Team.Components.html tiedoston teamlist *ngFor looppi (Reijonen, 2019)

4.1.3 Services

Servicen avulle voidaan jakaa dataa eri komponenttien välillä tehokkaasti. Service luokka voidaan injektoida komponentin constructoriin, jolloin datan jakaminen on helppoa. Service importataan ja injektoidaan projektin App.Component.ts tiedostoon, josta se voidaan jakaa eri komponentteihin käytettäväksi. Sovelluksen kaikki http kutsut tehdään service luokissa, jolloin sovelluksen toiminnallisuus on helpommin ymmärrettävissä ja kaikki http kutsut löytyvät samasta paikasta. Alla olevassa kuvassa (Kuva 8) on esimerkki Serviceluokan injektoinnista.

```
constructor(private service: PractiseService,
  private router: Router,
  private toaster: ToastrService) { }
ngOnInit() {
  this.refreshList();
}
```

Kuva 8. Esimerkki PractiseServicen injektoinnista toiseen .ts tiedostoon. (Reijonen, 2019)

Haluttu service voidaan tuota componenttiin importtaamalla se componentin Typescript tiedoston costructoriin. Yllä olevassa kuvassa Practiseservice ja ToastrService on tuotu practises.component.ts tiedostoon käytettäviksi.

4.2 ASP.NET Core 2.2

ASP.NET Core on avoimen lähdekoodin ohjelmistokehys, jonka on kehittänyt Microsoft. Ensimmäinen versio ASP.NET Coresta julkaistiin marraskuussa 2014 ja opinnäytetyössä käytetty versio 2.2 julkaistiin joulukuussa 2018. Projektissa Backendin kehittämiseen käytettiin C# - ohjelmointikieltä.

ASP.NET Coressa on seuraavat edut.

- ASP.NET Coressa on joukko arkkitehtuurisia muutoksia, jotka johtavat paljon kevyempään ja modulaarisempaan kehykseen.
- ASP.NET Core ei enää perustu System.Web.dll-tiedostoon vaan se on joukko NuGet-paketteja. Tämän avulla voit optimoida sovelluksesi sisältämään vain tarvitsemasi NuGet-paketit.
- Pienemmän sovelluksen etuihin kuuluvat tiukempi turvallisuus, parempi suorituskyky ja alennetut kustannukset
- Kyky hostata IIS: ssä.
- Rakennettu .NET Core, joka tukee todellista rinnakkaissovelluksen versiointia.

(ASP.NET Documentation)

4.3 GitHub

GitHub on verkkosivusto, joka tarjoaa paikan Git-versionhallintaa käyttäville ohjelmakehitysprojekteille. Git itsessään on komentoriviohjelma, jolle Github tarjoaa erään graafisen käyttöliittymän. Gitin lisäksi GitHub tarjoaa projekteille toimintoja kuten bugienseurannan, kehitystoiveet, tehtävien hallinta ja wiki.

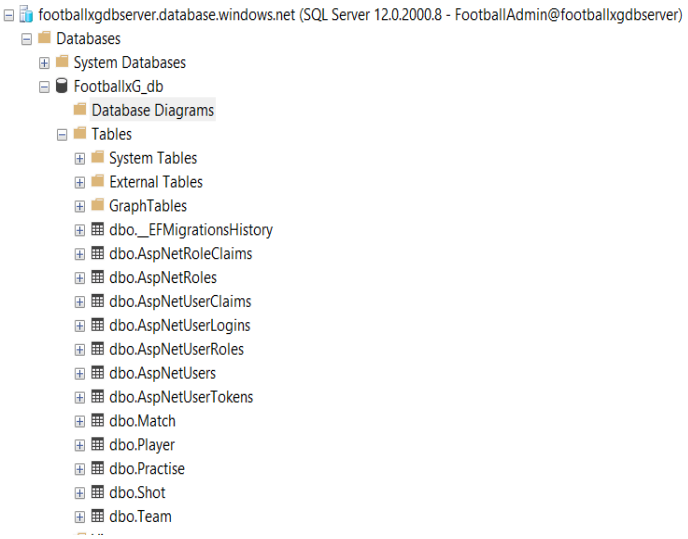
GitHub tarjoaa palvelunsa ilmaiseksi julkisesti nähtävillä oleville ohjelmavarastoille.

Huhtikuun 2016 GitHubin raportin mukaan sillä oli yli 14 miljoonaa käyttäjää ja yli 35 miljoonaa ohjelmavarastoa. Tämä tekee siitä maailman suurimman lähdekoodiverkkopalvelun. (Github Press Info. 2016.)

4.4 Microsoft SQL Server Management Studio

”SQL Server Management Studio (SSMS) on Microsoft SQL Server 2005:
n kanssa ensimmäisen kerran käynnistetty ohjelmistosovellus, jota käytetään kaikkien Microsoft SQL Server -komponenttien määrittämiseen ja hallintaan. SSMS sisältää sekä komentosarjojen editoijat että graafiset työkalut, jotka toimivat palvelimen objektien ja ominaisuuksien kanssa. ” (What is SQL Server Management Studio, Microsoft Docs). SSMS: n keskeinen ominaisuus on Object Explorer, jonka avulla käyttäjä voi selata, valita ja toimia minkä tahansa palvelimen objektin suhteen.

Football xG kehityksessä käytettiin SSMS SQLexpress palvelinta, joka tarjoaa käyttäjälle SQL palvelimen, mihin voidaan ottaa yhteys Visual Studio projektista ja täten luoda tarvittavia tietokantoja. Visual studio projekti ottaa yhteyden SQLexpress palvelimeen projektin Connection Stringin avulla, joka määritellään appsettings.json tiedostossa ja sieltä se syötetään projektin configuraatioon Startup.cs tiedostossa.



Kuva 9. SSMS managerin tietokanta näkymä FootballxG_db kannasta. (Reijonen, 2019)

Yllä kuva (Kuva 9) tietokanta näkymästä, kun SSMS on saanut muodostettua yhteyden Microsoft Azuren tarjoamaan SQL palvelimeen, jossa sijaitsee projektin tietokanta.

5 PROJEKTIN VAIHEISTUS

Projektin alkaessa pidettiin aloituspalaveri, jossa luotiin asiakkaan kanssa tekninen dokumentointi, johon listattiin sovellukselta vaadittavat ominaisuudet. Tämän jälkeen luotiin projektisuunnitelma ja projekti vaiheistettiin seuraavasti:

5.1 Suunnittelu

Ensimmäisenä toimenpiteenä hyväksytettiin projektisuunnitelma ja tekninen dokumentaatio asiakkaalta ja opinnäytetyöohjaajalta. Suunnitteluvaiheessa valitaan tietty määrä ominaisuuksista projektin backlogista ja hankitaan tarvittavat esitiedot näistä komponenteista. Lisäksi tutustutaan tarvittaviin manuaaleihin ja tietokanta pohjaan, joita saatetaan käyttää hyödyksi tulevan sprintin aikana.

5.2 Toteutus

Toteutus vaiheessa ohjelmaa kehitetään ja luodaan uusia komponentteja, jotka on valittu suunnitteluvaiheessa backlogista. Toteutus vaihe eli sprintin kehitys vaihe kestää kaksi viikkoa. Jos backlog saadaan tyhjennettyä kaikista sprintin ajalle suunnitelluista toteutuksista, valitaan lisää uusia kehitettäviä ominaisuuksia.

5.3 Testaus

Sprintin lopussa testataan valmiiksi saatuja ominaisuuksia ja raportoidaan lopputulokset tilaajalle ja opinnäytetyöohjaajalle. Tämän jälkeen tehdään korjauksia sovellukseen saatujen palautteiden pohjalta. Testauksen jälkeen siirrytään takaisin suunnittelu vaiheeseen ja aloitetaan seuraavan sprintin suunnittelu. Vaiheita toistetaan niin pitkään, kunnes sovellus on valmis eli arvioltaan viiden sprintin ajan. Kolmannen sprintin jälkeen sovellus julkaistiin Microsoft Azureen, ja asiakas pystyi testaamaan sovellusta oikeassa ympäristössä.

5.4 Lopetus

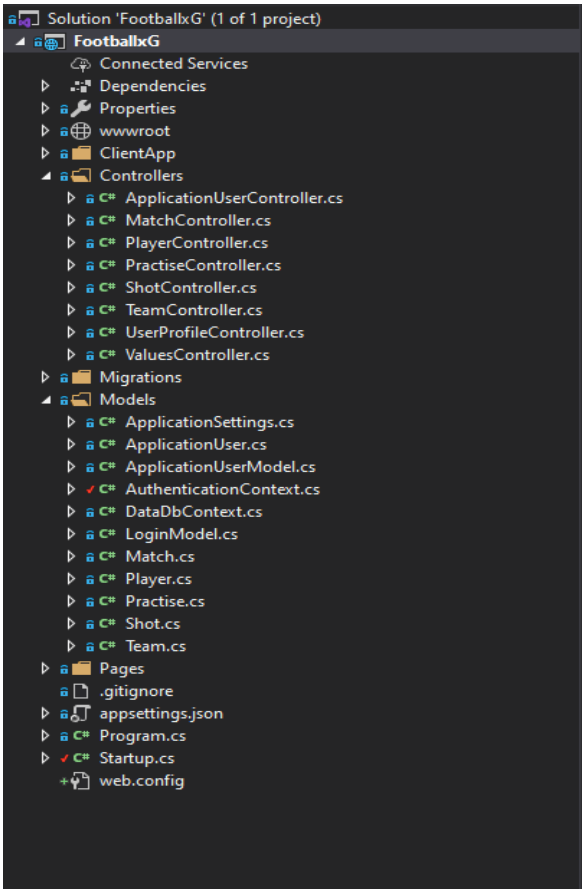
Kun kaikki aloituspalaverissa listatut osiot saatiin toteutettua ja sovellus oli testattu asiakkaan toimesta, siirryttiin projektin lopetus vaiheeseen. Tässä vaiheessa varmistettiin useaan kertaan sovelluksen toimivuus, Azure ylläpito, Azure maksusopimus ja laadittiin loppuraporttina tämä opinnäytetyö. Asiakkaalta kerättiin myös palautetta projektin onnistumisesta.

6 PROJEKTIN TOTEUTUS

Tässä luvussa käydään lävitse projektin vaiheistus eli missä järjestyksessä ja miten web-sovellus on toteutettu. Katsotaan lyhyesti läpi sovelluksen kansiorakennetta ja miten näihin ratkaisuihin päädyttiin.

6.1 Sovelluksen kansiorakenne

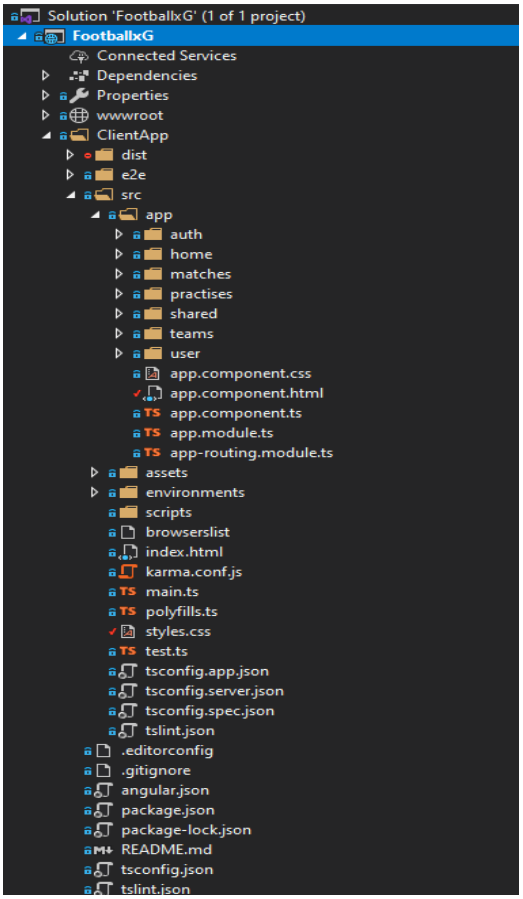
Sovellus on toteutettu käyttäen ASP.NET Core palvelinpuolta ja Angular8 Templatea käyttöliittymänä. Alla olevassa kuvassa (Kuva 10) nähdään sovelluksen palvelinpuoli.



Kuva 10. Sovelluksen palvelinpuolen näkymä. (Reijonen, 2019)

Controllereiden avulla hallinnoidaan käyttöliittymässä tehtyjä POST/GET/UPDATE/DELETE kutsuja. Kutsut menevät controllerin oikeaan funktioon ja toteuttavat oikeat toiminnallisuudet. Models luokkia käytetään controllereiden sisällä datan tallettamiseen oikeaan muotoon vastaamaan tietokannan tallennus vaatimuksia.

Alla olevassa kuvassa (Kuva 11) kansiorakenne sovelluksen FrontEndistä, joka löytyy projektin ClientApp kansiosta.

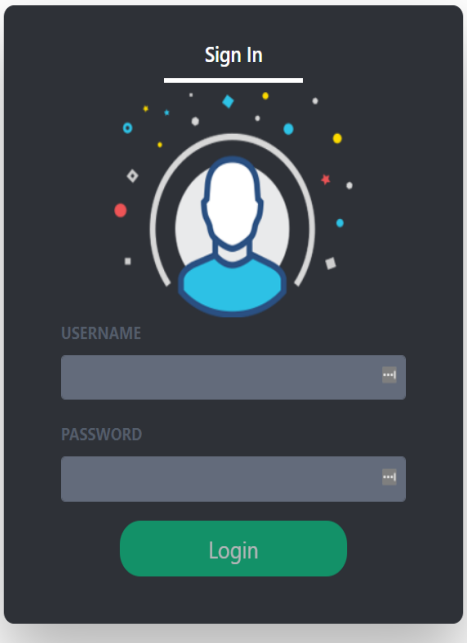


Kuva 11. Sovelluksen FrontEnd näkymä. (Reijonen, 2019)

ClientApp kansioon on generoitu pohja Angular 8 projektille. Frontendistä löytyy kirjautumissivu ”User”, kotisivu ”home” ja kolme alasivua ”practises, matches, teams”. Shared kansiosta löytyy sovellukselle yhteiset komponentit, joita ovat servicet ja modelit.

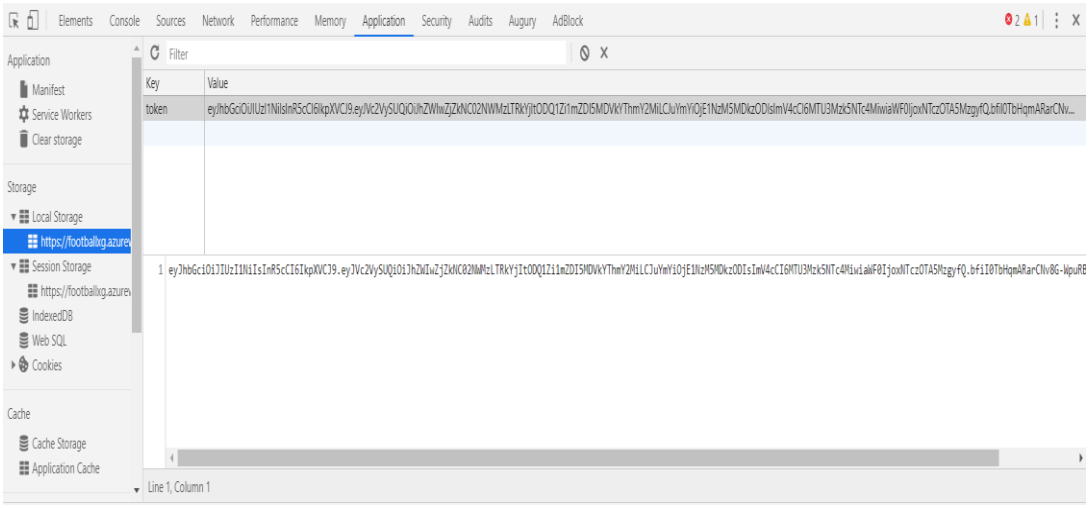
6.2 Sovelluksen toiminnallisuudet

Sovelluksen käynnistyessä, jos sovellus ei havaitse selaimen local storagessa JWT luomaa autentikointitokenia, ohjautuu käyttäjä sovelluksen kirjautumissivulle.



Kuva 12. Sovelluksen kirjautumissivu. (Reijonen, 2019)

Käyttäjä voi kirjautua sovellukseen syöttämällä käyttäjätunnuksen ja salasanan. Sovellus lähettää API kutsun, jossa se vertaa käyttäjän syöttämiä tunnuksia, jos tietokannasta löytyvät vastaavat tunnukset, pääsee käyttäjä autentikoitumaan sisään sovellukseen. BackEnd tallentaa autentikoinnin yhteydessä selaimen local storageen JWT tokenin.

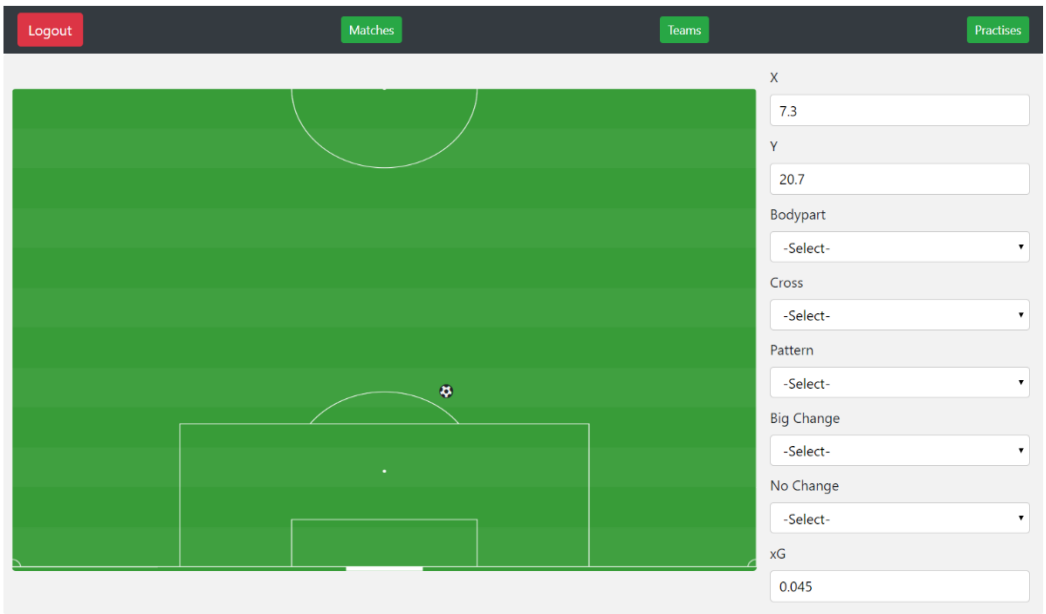


Kuva 13. Selaimen LocalStorageen tallennettu JWT tokeni. (Reijonen, 2019)

JWT token on salatussa muodossa selaimen local storagessa ja se voidaan purkaa vain BackEndistä löytyvällä secretkeyllä, jotta kuka tahansa ei pääse käsiksi käyttäjän tietoihin.

6.2.1 Kotisivu

Kirjautumisen jälkeen käyttäjä ohjautuu kotisivulle. Alla kuva kotisivusta.



Kuva 14. Sovelluksen home sivusto. (Reijonen, 2019)

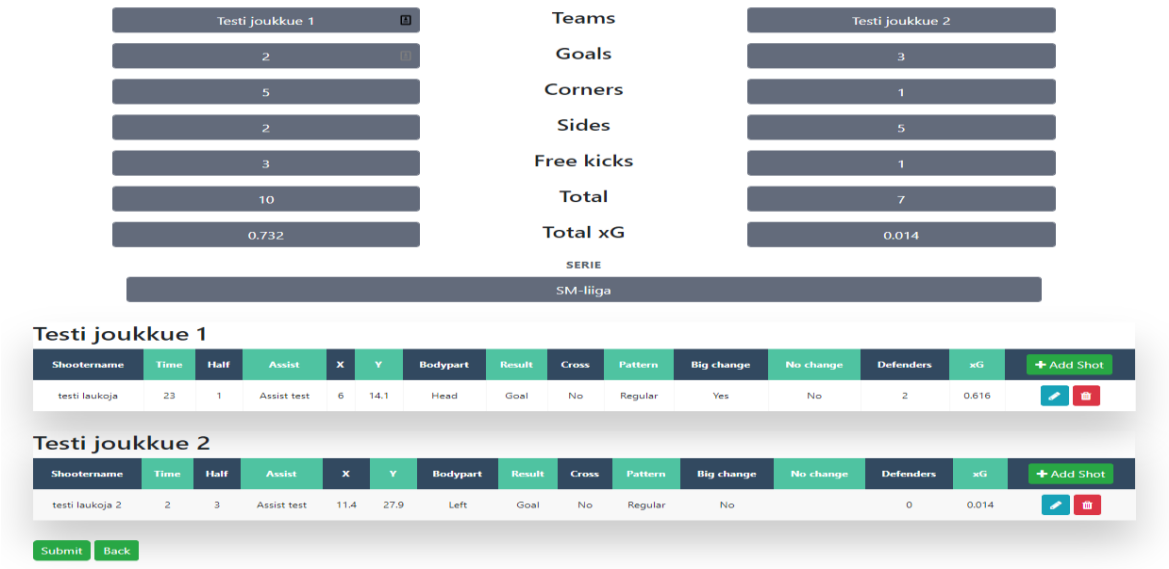
Kotisivustolta löytyy xG – laskentaan tarkoitettu laukaisukartta, jota painamalla sovellus laskee xG – dataa. Laskettua dataa ei tallenneta tietokantaan, koska tämä on tarkoitettu nopeita xG laskuja varten. Sivuston ylänavigoinnista löytyvät neljä alasivua ”Logout, Matches, Teams, Practises”.

Matches sivulta päästään luomaan ottelua tai tarkastelemaan entuudestaan luotuja otteluja. Teams sivulta löytyy luodut joukkueet ja pelaajat. Practises sivulta käyttäjä pääsee tarkastelemaan ja luomaan harjoituksia.

Punainen logout nappi ohjaa käyttäjän kirjautumaan ulos sovelluksesta. Nappia painamalla JWT token tuhotaan local storagesta ja käyttäjä ohjataan takaisin kirjautumissivulle.

6.2.2 Matches alasivu

Ottelut sivulla voidaan luoda tai katsastella aiemmin luotuja otteluita. Ottelulle voidaan asettaa kotijoukkue, sekä vierasjoukkue. Joukkueille voidaan syöttää tietoja otteluissa tehdyistä vedoista ja tallettamaan nämä tiedot tietokantaan. Alla kuva (Kuva 15) täytetystä ottelu sivustosta.



Kuva 15. Ottelun luonti alasivu. (Reijonen, 2019)

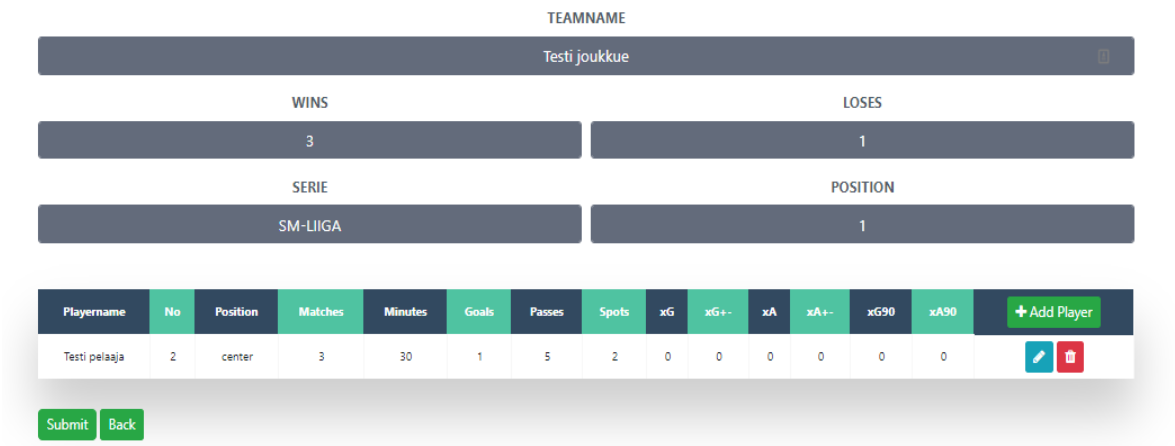
Otteluiden, sekä vetojen tekemiseen on käytetty CRUD menetelmää. Ottelun joukkueille voidaan myös lisätä dataa maaleista, kulmapotkuista, laitapotkuista, vapaapotkuista ja niiden yhteenlasketuista määristä. Luotujen potkujen yhteenlaskettu xG – data näkyy myös joukkueen tiedoissa. Add Shot painiketta painamalla aukeaa alla olevassa kuvassa oleva näkymä.

Kuva 16. Laukaisun luonti näkymä. (Reijonen, 2019)

Shot - näkymä toimii samalla tavalla, kuin etusivulla oleva laukaisukartta, mutta tässä näkymässä oleva data voidaan tallettaa tietokantaan.

6.2.3 Teams alasivu

Teams alasivulla voidaan luoda joukkueita tai muokata ennestään luotuja joukkueita. Joukkueille voidaan syöttää maali-, sekä pelaajadataa. Joukkueiden ja pelaajien luonti on toteutettu käyttäen CRUD menetelmää. Alla kuva luodusta joukkueesta ja joukkueeseen syötetyistä pelaajista



Kuva 17. Joukkueen luonti alasivu. (Reijonen, 2019)

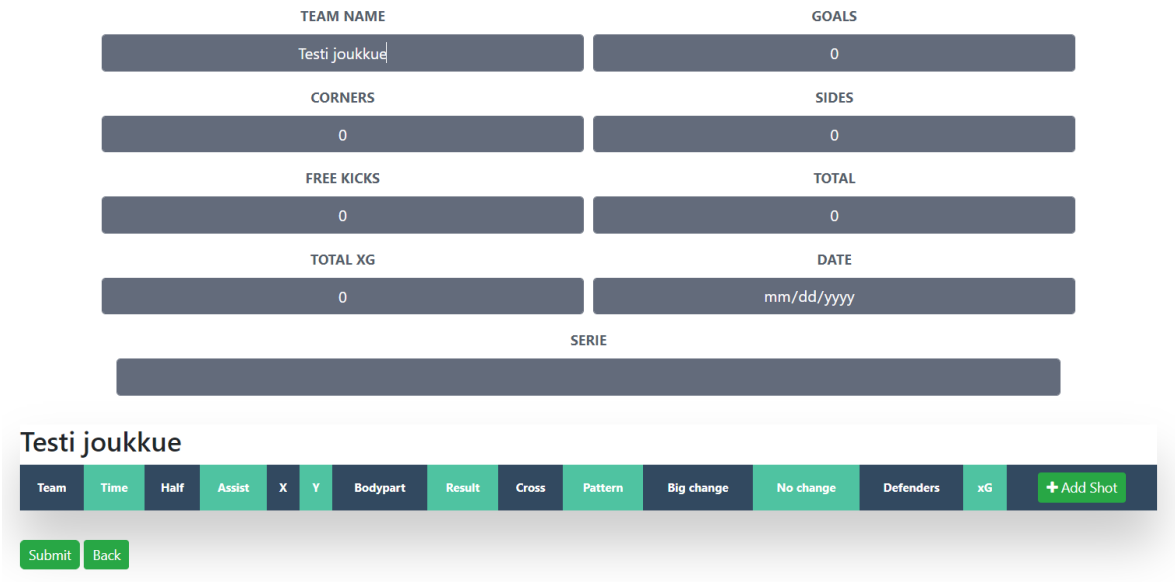
Joukkueelle voidaan perustietoina syöttää nimi, maalien-, häviöidenlukumäärät, sarjataso ja sijoitus sarjassa. Joukkueelle voidaan objektina syöttää lista pelaajista ja heidän henkilökohtaisesta datastaan. Uusi pelaaja luodaan Add Player napista, ja aiemmin luodun pelaajan dataa voidaan muokata painamalla haluttua riviä. Alla näkymä pelaajanluontinäkymästä.

Kuva 18. Pelaajan luonti joukkueelle. (Reijonen, 2019)

Näkymässä voidaan muokata yksinkertaisia inputteja. xG data päivittyy automaattisesti syötettyjen tietojen perusteella eikä näitä kenttiä käyttäjä voi itse muokata. Submit nappi lähettää uuden rivin pelaajalistaan mutta ei vielä tallenna mitään tietokantaan.

6.2.4 Practises alasivu

Practises alasivu on toteutettu samalla tavalla kuin Matches alasivu, mutta Practises sivustolla joukkueita on vain yksi. Tällä sivustolla joukkueelle voidaan luoda harjoituksia ja täten seurata pelaajien kehittymistä luomalla pelaajakohtaista xG –dataa. Alla kuva (Kuva 19) harjoituksen luomisesta ja luoduista testi/laukauksista.



Kuva 19. Harjoittelun luonti alasivu. (Reijonen, 2019)

7 SOVELLUKSEN JULKAISU

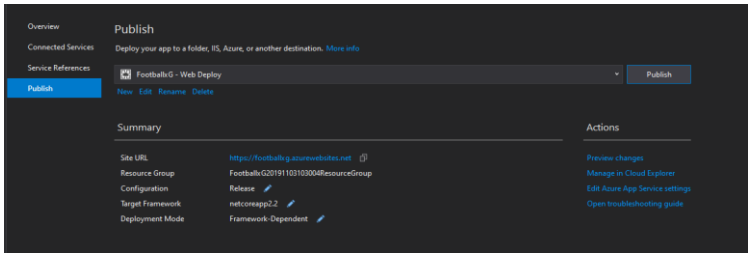
Aloituspalaverissa suunniteltiin, että valmis web-sovellus julkaistaan Microsoft Azureen asiakkaan käyttöön. Tässä luvussa käydään lävitse lyhyesti Microsoft Azuren toiminnallisuuksia ja julkaisun vaiheita.

7.1 Microsoft Azure

Microsoft Azure on Microsoftin tuottama avoin pilvipalvelu. Azurea voidaan käyttää sekä virtuaalipalvelinten alustana, että kehittäjille tarkoitettuna kehitysalustana. Microsoft Azure tarjoaa myös erilaisia valmiita pilvipalvelukomponentteja esimerkiksi mobiililaitteiden hallintaan, dokumenttien suojaamiseen, suurten datamassojen analysointiin ja koneoppimiseen.

7.2 Sovelluksen julkaisu

Azuren hinnoittelu määräytyy käyttäjän käyttötarpeen mukaan. Uusille rekisteröityneille käyttäjille se tarjoaa 30-päivän ilmaisen kokeilun, ja tämän jälkeen 170 euroa ilmaista kokeilua varten. Football xG käyttötarkoitukseen tarvitsimme App Servicen, tietokannan ja SQL palvelimen. Sovelluksen julkaiseminen tapahtuu valitsemalla Visual Studio projektista publish osio ja valitsemalla New Profile. Tämän jälkeen Visual Studio yhdistää Microsoft Azureen ja luo valitut toiminnot. Alla kuva (Kuva 19) luodusta Publish Profilesta.



Kuva 20. Visual studio projektin publishaaminen Microsoft Azure App Serviceen. (Reijonen, 2019)

Microsoft Azure luo automaattisesti uuden Connection stringin jonka avulla voimme ottaa yhteyden SSMS:n avulla Azuren SQL palvelimeen ja luoda uusilla migraatioilla tietokanta taulut.

8 YHTEENVETO JA JATKOKEHITYSIDEAT

Tässä luvussa pohditaan annettuihin tavoitteisiin pääsemistä, projektin aikataulutusta, mitä opittiin, mitkä asiat olivat helppoja, oliko asiakas tyytyväinen lopputulokseen, sekä mitkä asiat tuottivat vaikeuksia. Lopussa käsitellään jatkokehitysideoita.

8.1 Yhteenveto

Projektin tavoite oli tehdä asiakkaalle moderneilla web-kehityksen teknologioilla kehitetty web-sovellus, jolla käyttäjät voivat laskea, tallentaa, poistaa, muokata ja katsella xG-dataa.

Projektin aloituspalaverissa listatut tavoitteet toteutettiin dokumentoinnin mukaan, ja vaaditut tavoitteet saavutettiin onnistuneesti. Huomioitavaa on myös, että sovellus toimii moitteettomasti kaikilla laitteilla ja selaimilla, lisäksi sovellus skaalautuu laitteen mukaisesti.

Projektille määriteltiin pitkä aikataulu, koska työskentelen täysipäiväisesti, joten pystyin kehittämään sovellusta pääsääntöisesti vain viikonloppuisin. Asiakkaalle tämä aikataulu oli myös sopiva, koska sovelluksen käyttöönotolla ei ollut kiirettä. Laaditusta aikataulusta onnistuttiin pitämään kiinni onnistuneesti, ottaen huomioon, että sovellusta kehitettään vain viikonloppuisin. En kuitenkaan ole täysin tyytyväinen sovelluksen lopputulokseen, koska sovelluksen graafinen ulkoasu ei ole mielestäni hyvällä tasolla. Minulla, sekä asiakkaalla ei ole kokemusta ulkoasun suunnittelusta, joten ulkoasun suunnittelu jäi minimaaliseksi. Jatkokehityksessä sovelluksen ulkoasuun panostettaisiin enemmän.

Projektissa ei tullut minulle paljoakaan uusia teknologioita opittavaksi. Olen aiemmin työelämässäni käyttänyt Angular8, ASP.NET Corea, GitHubia, sekä Microsoft Azurea. Uutena asiana minulle tuli projektin julkaiseminen Microsoft Azure ympäristöön. Sovelluksen julkaisemiseen löytyi Microsoft Azuren sivustolta hyvät dokumentaatiot eikä julkaiseminen loppujen lopuksi tuottanut suuria ongelmia. Olen

aiemmissa projekteissa suunnitellut sovelluksen arkkitehtuuria, mutta en ikinä näin isossa muodossa. Sovelluksen arkkitehtuurin suunnitteluun etsin vastaavanlaisia projekteja luin näiden projektien dokumentaatiota, sekä arkkitehtuuria, joista sain ideoita Football xG:tä varten.

Sovellusten suunnittelusta ja tekemisestä minulla oli jo aikaisempaa kokemusta, joten suunnitteluvaihe meni projektissa sulavasti. Angular8 ja ASP.NET Core projektien luominen Visual Studiolla olivat minulle ennestään tuttua. Lisäksi minulta löytyi vankkaa FrontEnd osaamista, joka helpotti sovelluksen kehitystä, koska sovellus oli suurimmalta osaltaan FrontEnd kehitystä.

8.2 Jatkokehitysideat

Projektin aikana pidetyissä kehityspalavereissa tuli ilmi lukuisia jatkokehitysideoita, joita ei oltu projektinaloituspalaverissa listattu sovelluksen vaadittuihin toimintoihin, ja asiakas osoitti innostusta sovelluksen tuotteistamisesta. Sovelluksen tuotteistaminen vaatisi kaikkien GDPR-lakien ja tietosuojavaatimusten läpikäymistä, koska sovelluksessa käsitellään henkilötietoja kirjautumisen yhteydessä, pitää nämä tiedot olla oikein salattuja ja turvattuja.

Jatkokehityksessä lähdettäisiin hakemaan ulkopuolista apua sovelluksen graafisen suunnittelun suhteen. Nykyisessä tilassaan sovellus toimii moitteettomasti ja toteuttaa kaikki aloituspalaverissa kirjatut toiminnot, mutta graafiseen ulkoasuun ei ole panostettu riittävästi.

9 LÄHTEET

Angular Documentation [verkkoaineisto]. [Viitattu 2019-09-12.]
Saatavissa: https://angular.io/

ASP.NET Core Tutorial [verkkoaineisto]. [Viitattu 2019-07-12.] Saatavissa: https://www.tutorialspoint.com/asp.net_core/asp.net_core_overview.htm

ASP.NET Team. Introduction to ASP.NET Identity [verkkoaineisto]. [Viitattu 2019-07-12.] Saatavissa: https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity

Auth0. Introduction to JSON Web Tokens[verkkoaineisto]. [Viitattu 2019-05-12.]
Saatavissa: https://jwt.io/introduction/

Azure get started [verkkoaineisto]. [Viitattu 2019-05-12.]
Saatavissa: https://azure.microsoft.com/en-us/

Calandra, Mariano. Why do we need the JSON web token (JWT) in the Modern Web Era? [verkkoaineisto]. [Viitattu 2019-03-12.] Saatavissa: https://hackernoon.com/why-do-we-need-the-json-web-token-jwt-in-the-modern-web-k29l3sfd

Carragher, Jamie and Jones, David. Expected goals explained [verkkoaineisto]. [Viitattu 2019-03-12.] Saatavissa: https://www.skysports.com/football/news/11096/10989764/monday-night-football-extra-expected-goals-explained

GIthub Press Info[verkkoaineisto]. [Viitattu 2019-08-12.]
Saatavissa: https://github.com/about/press

Heller, Martin, REST and CRUD: The Impedance Mismatch [verkkoaineisto]. [Viitattu 2019-08-12.] Saatavissa: https://www.infoworld.com/article/2640739/rest-and-crud--the-impedance-mismatch.html

Hyppänen, Antti. Vedonlyöntiopas: Jalkapallon tilastointi [verkkoaineisto]. [Viitattu 2019-01-12.] Saatavissa: https://asialinja.com/vedonlyontiopas-jalkapallon-tilastointi/

Javatpoint team. Angular 7 tutorial [verkkoaineisto]. [Viitattu 2019-01-12.] Saatavissa: https://www.javatpoint.com/angular-7-tutorial

JSON Web Token [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: https://fi.wikipedia.org/wiki/JSON_Web_Token

Koretskyi, Max. A plain English introduction to JSON web tokens (JWT): what it is and what it isn't [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: https://medium.com/ag-grid/a-plain-english-introduction-to-json-web-tokens-jwt-what-it-is-and-what-it-isnt-8076ca679843

Ohjelmointirajapinta wikipedia [verkkoaineisto]. [Viitattu 2019-07-12.] Saatavissa: https://fi.wikipedia.org/wiki/Ohjelmointirajapinta

Page, Danny. Expected Goals Just Don't Add Up – They Also Multiply [verkkoaineisto]. [Viitattu 2019-07-12.] Saatavissa: https://medium.com/@dannypage/expected-goals-just-don-t-add-up-they-also-multiply-1dfd9b52c7d0

Pisuwala, Ubaid. Angular 7 – What are the new improvements and features [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: https://www.peerbits.com/blog/angular-7-features-and-updates.html

Sumpter, David. The Geometry of Shooting [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: https://medium.com/@Soccermatics/the-geometry-of-shooting-ae7a67fdf760

Tietoliikenneohjelmointi Oamk [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: http://www.oamk.fi/~eniemi/TietolOhj/materiaali/TietoliikenneohjV004.pdf

What is ASP.NET Core? [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core

What is SQL Server Management Studio (SSMS)? [verkkoaineisto]. [Viitattu 2019-09-12.] Saatavissa: https://docs.microsoft.com/fi-fi/sql/ssms/sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15