Rekstrarhandbók

FramtíðarSTEF

Eva Sif Einarsdóttir

Hreiðar Kristinn Hreiðarsson

Ísabella Ýr Finnsdóttir

Jóhanna María Gísladóttir

Unnur Lára Halldórsdóttir

Júní 2019

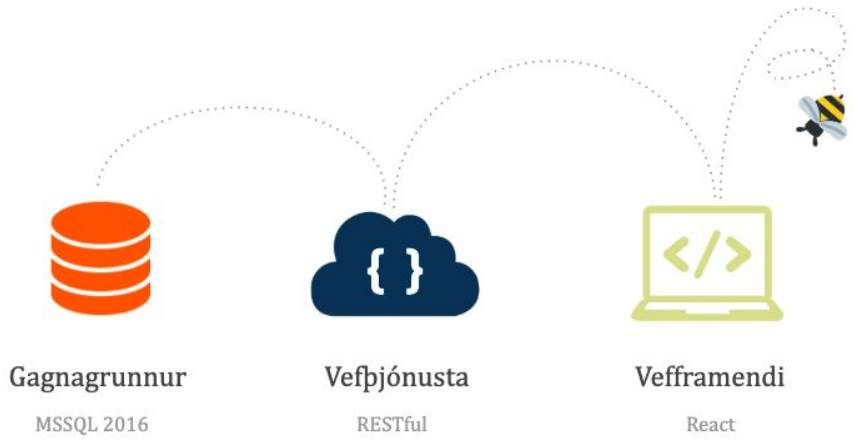
Efnisyfirlit

1 Inngangur

Þessi handbók er fyrir rekstur FramtíðarSTEFs kerfisins. Þessi handbók mun veita stjórnendum og þeim sem munu koma að frekari þróun kerfisins upplýsingar um hvernig eigi að nálgast verkefnið.

2 Kerfið

Kerfið er vefsíða sem nýtir sér RESTful vefþjónustu til þess að sækja öll gögn STEFs úr gagnagrunni þeirra sem er hýstur hjá íslensku upplýsingatækni fyrirtæki. Gagnagrunnur kerfisins er í MSSQL 2016 og inniheldur örfáar stefjur sem nýttar eru af vefþjónustu kerfisins. Stefjurnar eru fyrir stærstu og mikilvægustu aðgerðir kerfisins og aðstoða vefþjónustuna við að framkvæma þær. Vefþjónustan sem kerfið nýtir sér er RESTful þar sem allir endapunktar eru vel skilgreindir og CI ferli er fléttað þar inn í til þess að kerfið sé útgefanlegt (*e. Deployable*). Vefþjónusta er skrifuð í .NET Core og C# forritunarmálinu. Framendi kerfis er skrifaður í React í forritunarmálinu JavaScript, en framendinn styðst við vefþjónustuna.



3 Þróunar kröfur

Eftirfarandi skref eru nauðsynleg til að hefja þróun á verkefninu.

3.1 Uppsetning

Þetta kerfi er háð því að *Git*, *Node*, *Node Package Manager(NPM)* og *Dotnet* sé uppsett á tölvunni sem keyrir verkefnið. Einnig þarf að hafa aðgang að gagnagrunni, en hægt er að fá aðgang að honum með því að hafa samband við STEF.

- Hægt er að ná í *Git* hér: https://git-scm.com/downloads
- Hægt er að ná í *Node* og *NPM* hér: https://www.npmjs.com/get-npm
- Hægt er að ná í Dotnet hér: https://dotnet.microsoft.com/download

```
# Klóna verkefni frá git
git clone https://gitlab.com/stef-is/framtidarstef.git

# Vefþjónusta
# Fara inn í framtidarstef möppuna
cd framtidarstef

# Fara inn í framtidarstef-api möppuna
cd framtidarstef-api

# Fara inn í FramtidarSTEF.WebApi möppuna
cd FramtidarSTEF.WebApi

# Endurbyggja vefþjónustu
dotnet restore

# Byggja og keyra vefþjónustu á eigin vél, keyrir á http://localhost:5000
dotnet build && dotnet run

# Prófanir
# Fara til baka í framtidarstef-api
cd ..

# Keyra prófanir
dotnet test

# Framendi
# Opna nýjan flipa fyrir framenda möppuna

# Fara til baka í framtidarstef möppuna
cd ..

# Fara inn í framtidarstef-client möppuna
cd framtidarstef-client

# Setja upp pakka sem þarf
npm install

# Keyra upp framenda á eigin vél, keyrir á http://localhost:9000 */
npm start
```

Mynd 2.1: Uppsetning

4 Kóða reglur

Notast er við Prettier pakkann í Visual Studio Code til þess að hafa samræmi í kóðanum en Prettier umbreytir kóðanum samkvæmt þar til gerðum kóðareglum. Ásamt því að nota Prettier fylgdi teymið eigin kóðareglum sem settar voru í upphafi verkefnisins til þess að auka samræmi og snyrtileika kóða enn frekar. Hér að neðan má sjá þær reglur sem fylgt var.

4.1 C#

- 2 bil fyrir inndrátt
- camelCase fyrir nöfn á breytum
- PascalCase fyrir nöfn á föllum
- SNAKE_CASE í hástöfum fyrir nöfn á breytum í módelum
- Engin auka bil
- Kommenta kóðann þar sem þarf
- Ein tóm lína á milli falla
- Gagnagrunns-fyrirspurnir í hástöfum (SQL skipanir)
- 3 laga hönnun
- Nota skil (*e. Interfaces*)
- Ákvæð innspýting (*e. Dependency injection*)

4.2 JavaScript

- Fylgja Eslint reglum
- Röðunarreglur eftir JSX málskipan (*e. Syntax*)
- Tög lokast sjálf (*e. Self-closing tags*)
- camelCase fyrir nöfn á breytum og föllum
- Engin auka bil
- Kommenta kóðann þar sem þarf

- 2 bil fyrir inndrátt
- Ein tóm líka á milli falla
- Öllum þáttum (*e. Elements*) verður að vera lokað
- Gildi sem að eru gefin eigindum (*e. Attributes*) verða að vera í gæsalöppum

4.3 CSS

- Nota klasa fyrir veljara (*e. Selectors*) en forðast að nota id til fyrir betri endurnýtingu á kóða
- Aðeins nota ensk orð
- camelCase fyrir veljara (*e. Selectors*)
- Klasa nöfn eiga að vera lýsandi
- Forðast auka veljara (*e. Selectors*)

5 Uppbygging verkefnis

5.1 Vefþjónusta

Vefþjónustunni er skipt í þrjú lög og fylgir *RESTful* staðlinum. Neðsta lagið er gagnalagið (*e. Repository layer*) en þar er náð í gögnin frá gagnagrunninum með *Dapper ORM*. *Dapper* er ormur sem sendir SQL fyrirspurnir á gagnagrunninn og skilar objectum til baka. Þaðan eru gögnin send í miðjulagið sem kallast þjónustulag (*e. Service layer*) en í þjónustulaginu fara fram allir útreikningar og villutékk. Gögnin eru send frá þjónustulaginu upp í efsta lagið, afhendingarlagið (*e. Presentation/WebApi layer*) en þaðan er hægt að nálgast og senda inn gögn á þægilegan hátt.

Einingaprófanir eru gerðar í vefþjónustunni þar sem MSTest og Moq eru notuð til þess að prófa kóðann. Pakkinn Coverlet er notaður til þess að gefa út „code coverage" skýrslu um það hversu mikið af kóðanum er prófaður. Mikilvægt er að kóðinn sé prófaður og markmiðið er að halda honum prófuðum í 70%. Þessir prófunarpakkar eru fyrir .NET Core en vefþjónustan er smíðuð í því.

5.2 Framendi

Í framendanum er kallað beint á afhendingarlagið (*e. Presentation/WebApi layer*) úr Redux geymslunni til þess að senda gögn í vefþjónustuna og ná í gögn úr henni. *Redux* er stöðu geymsla (*e. State container*) fyrir *JavaScript* forrit. *Redux* vinnur mjög vel með *React* og getur hjálpar mikið við að stjórna heildar stöðu *(e. Global state*) forritsins. Einnig er öllum kóða í framendanum skipt niður í viðeigandi hluta (*e. Component*) eftir virkni þeirra. Framendinn er prófaður í Cypress, en það er kóðaprófunarkerfi fyrir JavaScript forritunarmálið sem framendinn er þróaður í.

5.3 Uppbygging mappa

framtidarstef

- |-framtidarstef-api
  - |-FramtidarSTEF.Models
  - |-FramtidarSTEF.Repositories
  - |-FramtidarSTEF.Services
  - |-FramtidarSTEF.Tests
  - |-FramtidarSTEF.WebApi
- |-framtidarstef-client
  - |-src
    - |-actions
    - |-components
    - |-constants
    - |-helpers
    - |-images
    - |-reducers
    - |-services
- |-scripts

6 Útgáfa (deployment)

FramtíðarSTEF er uppsett með samfelldri samþættinga (*e. Continuous Integration*) pípu. Samþættinga pípa getur flýtt fyrir þróun á nærri öllum hugbúnaðarverkefnum, bætt gæði og aukið sjálfstraust þeirra sem koma að þróun hugbúnaðarins. Í þessum kafla verður farið yfir hvernig samfellda (*e. Continuous*) vinnuferlinu var háttað við þróun FramtíðarSTEFs.

6.1 Gitlab

Allur kóði er hýstur á útgáfustjórnunarkerfinu Gitlab þar sem samfelldur dreifingarþjónn (*e. Continuous deployment server*) er innbyggður og nýttur í CI ferlinu. Í hvert skipti sem ný breyting kemur inn á Gitlab gagnahirslu (*e. repository*) verkefnisins fer útgáfupípa sjálfkrafa í gang. Ef eitthvað stig í pípunni tekst ekki munu eftirfarandi stig ekki keyra og eru þau merkt sem bilun. Stigin sem pípan fer í gegnum er hreinsunarstig, byggingarstig vefþjónustu, byggingarstig framenda, prófunarstig og útgáfustig.

6.2 Hreinsunarstig (*e. Clean stage*)

Þetta er fyrsta stigið í samfelldu útgáfuferli (*e. Continuous Deployment*) pípunnar inni á Gitlab. Í þessu ferli þá hreinsar git upp allar þær skrár sem eru ekki hluti af kóðastjórnun og eyðir út öllum breytingum sem hafa ekki verið færðar inn á Gitlab.

6.3 Stig til þess að smíða vefþjónustu (*e. Build-api stage*)

Þetta er annað stigið í útgáfuferlinu þar sem vefþjónustan er hreinsuð, smíðuð og keyrð upp með því að nota dotnet frá Microsoft. Þetta stig notast við dotnet *restore*, *build* og *run*. Á milli hreinsunar- og smíðastigs þá eru ýmsar skriftur keyrðar upp til þess að uppfæra pakka á sýndarvél frá DigitalOcean. Í þessu stigi er svokölluð Docker mynd (e. image) nýtt

við að byggja allan kóða eftir breytingu, þar sem myndin er undirbúin fyrir útgáfu kerfisins. Þetta er mikilvægt skref í ferli pípunnar til þess að geta gefið kerfið út.

6.4 Stig til þess að smíða framenda (*e. Build-client stage*)

Þetta er þriðja stig í útgáfuferlinu þar sem veframendinn er keyrður upp með npm. Þetta stig er mikilvægt til þess að notendur geti séð allar þær upplýsingar sem þeir þurfa. Þetta stig notast við npm install og npm start. Í þessu stigi er svokölluð Docker mynd (*e. image*) nýtt við að byggja allan kóða eftir breytingu, þar sem myndin er undirbúin fyrir útgáfu kerfisins. Til þess að geta gefið kerfið út þá er þetta mikilvægt skref í ferli pípunnar eins og í stiginu að undan.

6.5 Prófunarstig (*e. Test stage*)

Prófunarstigið er fjórða stigið í útgáfuferlinu, þar sem keyrð eru bæði einingaprófanir og próf til að athuga hversu mikið af kóðanum er prófaður (*e. Code Coverage*). Í kerfinu er .sln skrá þar sem einingaprófanir eru keyrðar upp með skipuninni dotnet test. Prófunarferlið gengur úr skugga um að engar breytingar brjóti í bága við kröfur kerfis og ef það kemur fyrir má sjá á Gitlab ef pípan klikkar (*e. fails*).

6.6 Útgáfustig (*e. Deployment stage*)

Útgáfustigið er fimmta og síðasta stigið í útgáfuferlinu. Til að byrja með þá er Docker myndin byggð og henni ýtt (*e. push*) með Docker. Myndin er merkt með commit streng frá Gitlab, sem gerir Docker kleift að keyra rétta mynd upp. Þetta gerist á undirbúningsstigi (*e. staging*) en á framleiðslustigi (e. production) þá er notað Docker compose sem keyrir myndina okkar upp eftir stillingum sem eru skilgreindar í docker-compose.yml skránni.

7 Vefsíða

Vefsíðan keyrir ofan á vefþjónustunni, en bæði vefþjónustan og vefsíðan eru hýst á *DigitalOcean* sýndarvél. Gagnagrunnurinn er hýstur á vél hjá öðru fyrirtæki og nær vefþjónustan í gögnin þaðan. Til þess að nota þessa vefsíðu þá þarf notandi að vera með notendanafn og lykilorð sem að hægt er að fá hjá STEF. Ef tenging við gagnagrunn næst ekki þá getur notandi ekki komist inn í kerfið.

Þegar breytingar eru gerðar á kerfi þá þarf að færa þær inn á Gitlab aðgang STEFs. Þá fer CI ferlið sem fylgir kerfinu í gang og því mikilvægt að vera með aðgang að Gitlab. Eftir það ferli og útgáfu á DigitalOcean vélina á vefsíðan að uppfærast og þær breytingar sem að hafa verið gerðar á kerfinu eiga að vera sjáanlegar á vefsíðunni á netinu.