3. ÚTGÁFA | FEBRÚAR 2025

Þróunarhandbók

Leiðarvísir fyrir hugbúnaðarþróun fyrir embætti landlæknis

Embætti landlæknis

Þessi þróunarhandbók hefur að geyma leiðbeiningar, almennt verklag og leiðarvísa sem snúa að þróun, gæðaeftirlits og útgáfu hugbúnaðar fyrir embætti landlæknis Íslands.

Athugið að efni þessa skjals ber að líta á sem **leiðarvísi** til að hjálpa embættinu (okkur) að ná sínum langtíma markmiðum í hugbúnaðargerð. Tilgangur þessa skjals er ekki að vera reglulisti né nær það til allra hluta í hugbúnaðargerð og rekstri. Því eru allir sem lesa þennan texta **eindregið hvattir** til að hugsa krítískt um efni skjalsins, koma með **tillögur að úrbótum** og benda á **villur**, **veikleika** eða **vankanta annað hvort beint við verkefnastjóra okkar eða** á mrh@landlaeknir.is

Fyrirvari

Lausnir í lausnabanka sem og meðfylgjandi þróunarhandbók er varpað fram eins og þær koma fyrir, án hvers kyns ábyrgðar á hugsanlegum villum sem kunna að leynast í þeim lausnum eða í meðfylgjandi þróunarhandbók sem sett var saman til notkunar innan embættis landlæknis.

Í engu tilfelli er hægt að draga til ábyrgðar embætti landlæknis eða höfunda lausnanna/þróunarhandbókar eða beina til þeirra kröfum, höfða dómsmál eða annars konar ábyrgð enda eru fyrrgreindar lausnir ekki til þess fallnar að notaðar séu óbreyttar í lausnir verksala fyrir embætti landlæknis og er þróunarhandbókin eingöngu leiðarvísir fyrir hugbúnaðarþróun innan embættis landlæknis.

Hvers kyns notkun á fyrrgreindum lausnum embættisins og nefndu fylgiskjali, eins og þær koma fyrir við afhendingu til verksala, er á ábyrgð verksala sjálfs og embætti landlæknis að skaðlausu.

Þróunarhandbók og lausnir í lausnabanka er eign embættis landlæknis og er frekari birting óheimil nema með samþykki embættis landlæknis.

Breytingar frá fyrri útgáfu

Nýtt efni

- Bætt við lagalegum fyrirvara.
- Bætt við athugasemd um lög og fyrirmæli um skráningu í sjúkraskrám.
- Gæða og útgáfuferlar: Bætt við grein um ábyrgðir og væntingar
- Rekjanleikakröfur í skilum: Bætt við útskýringu og skýringarmynd á hverskonar rekjanleiki þarf að vera í skilum milli kóðagreina, PR, gæðaskýrsla, prófunarniðurstaða o.þ.h.
- Upplýsingum um lausnarbanka bætt við.
- Umræða um róttæka einföldun bætt við.
    - Upplýsingum um viðmiðunararkitektúr fyrir róttæka einföldun bætt við.
- Kröfur um þolgæði lausna bætt við.
- Athugasemdir um virknisflögg (e. feature flagging).

- Nýr kafli um Öryggi og auðkenningar.
- Nýr liður um æskilegan tæknistafla
- Framendar: Frekari útskýringar á einföldun framendastafla.
- Miðlun gagna: Kröfur um dulkóðun skýrðar, kröfur um útgáfustjórnun á gagnagrunnskóðar skýrðar.
- Geymsla leyndarmála í keyrsluumhverfi skýrðar og viðauka tengdu því bætt við.
- Rekstur og eftirlit: Kröfur til þjónustusamninga (SLA) bætt við.
- Viðauki 1: Fyrirkomulag sjálfvirknivæðingar MRH vs verktaki skýrt og hvar innviðakóði ætti að geymast. „Mikilvæg atriði“ uppfærð með nýjum punktum.
- Viðauki 2: Endurskrifaður, upplýsingar um kröfur til .NET lausna vs. javascript lausna skýrðar.
- Viðauki 4: Gátlistar uppfærðir og nýjir gátlistar fyrir útgáfu hugbúnaðar og verklok bætt við.
- Viðauki 5: Leiðbeiningar varðandi kóðarýni bætt við.
- Viðauki 6: Útgáfudagatal bætt við.

Breytingar

- Kóðageymslur: Viðbótar upplýsingar sem þarf þegar ný kóðageymsla er stofnuð.
- Kóðageymslur: Viðbótar dæmi um kóðageymslunöfn.
- Fork-and-branch: Aðlögun á verkferli til að hægt sé að vinna betur með Github Enterprise og Private Forkum.
- Fork-and-branch: Ekki er lengur gerð krafa um að merge commit séu squashed.
- Nafnareglur kóðagreina: Nákvæmara dæmi um nafnareglur, viðbótarkröfur fyrir QA kóðagreinar.
- Gæðastýringaferlar: Gert skýrt að þróunarhandbók er m.a. lögð til grundvallar í kóðarýnum.
- Hvenær er æskilegt að skila inn: Skilavæntingar í test umhverfi uppfærðar.
- Forritunarmál og framework: Skýrt að notast skuli við ASP.NET í framendalausnum. Skýringarmynd á hvað hybrid lausn er bætt við. Staða embættis gagnavart javascript SPA framendum gert skýrt.
- Athugasemd um caching notkun bætt við í þjónustu umræðu.
- Einingaprófanir: Nafnareglur einingaprófana skýrðar og vísanir í kröfur til þeirra.
- Endurnýting kóða: Kröfur tengdar einkasöfnum eða -pökkum skýrðar. Kröfur á kóðaskilum fyrir slíkan kóða skýrðar.
- Hugbúnaðarskjölun: Skýringar tengdar hvenær leyfilegt er að skila inn Markdown gögnum.
- Hugbúnaðarskjölun: Fjarlægði hluta tengdum CI skjölun.
- Gagnagrunnar: Skýringar á stefnum okkar frá Oracle og yfir í MSSQL, nánari upplýsingar um hýsingarumhverfi.
- Mælingar: Kröfur um ákveðið port fyrir /metric endapunkta fjarlægðar.

Efnisyfirlit

Áskorunin okkar

Hvað er verið að reyna að leysa

Við miðum við að hugbúnaðarlausnir sem smíðaðar eru fyrir okkur verði í rekstri og viðhaldi í allt að 30 ár. Langtímasýn er nauðsynleg því að á þessum 30+ ára rekstrar og viðhaldstíma er gert ráð fyrir að mörg hugbúnaðarteymi komi að verkefnunum og geri á þeim viðbætur og lagfæringar. Yfir 30 ára tímabil verða einnig miklar tækniframfarir og margar sveiflur í straumum og stefnum. Því leggjum við áherslu á að við hugbúnaðargerð séu ákvarðanir teknar sem miða að því að arkitektúr og tæknistafli standist til lengri tíma litið.

Það sem við leggjum sérstaklega áherslu á er

- Framtíðartryggðar tækniákvarðanir
- Viðhaldsþolinn hugbúnaðararkitektúr
- Skýr og einföld sjálfvirknivæðing
- Kjarnyrt skjölun
- Að lögum og fyrirmælum um skráningu í sjúkraskrám sé framfylgt

Framtíðartryggðar tækniákvarðanir

Það er ómögulegt að vita hvað framtíðin ber í skauti sér. Það er samt mögulegt og á sama tíma mikilvægt að reyna eftir bestu getu að lágmarka kostnað og “spól í sömu förum” tengdum breytingum á tækniumhverfi yfir lengri tíma.

Eftirfarandi spurningar þarf að hafa í huga þegar tekin er ákvörðun um notkun á tækni í bæði ný og eldri verkefni

1. **Bakhjarlinn** Hverskonar bakhjarl hefur tæknin og hversu líklegt er að bakhjarlinn fjárfesti í henni til langtíma?
    a. Er tæknin ný vara og líkleg til að taka miklum breytingum? T.d. er henni viðhaldið af hópi áhugafólks eða af fámennum hópi innan nýs fyrirtækis?
    b. Hvert er orðspor bakhjarlsins, fjárfestir hann almennt til langtíma í tækni eða eru teknar ófyrirséðar ákvarðanir um að hætta að styðja tækni hjá þeim?
2. **Tæknin** Er forritasafnið (e. library) eða ramminn (e. framework) sem verið er að velja í lausnina þroskað? Hversu líklegt er það til að vera enn þá til staðar og notað eftir 5 ár? En eftir 10 ár?
    a. Samræmist tæknin öðrum verkefnum hjá okkur? Er þetta gjörólíkt því sem þegar er til staðar og þekking er á?
    b. Krefst tæknin þess að nota þarf nýtt forritunarmál?
    c. Er tæknin hönnuð fyrir þær kröfur sem við höfum til hugbúnaðar? Er tækninni ætlað að leysa vandamál sem einkenna stærri markaði/hópa en eiga lítið eða ekkert við kröfur smærri markaðs eins og Íslands?
    d. Fellur tæknin inn í sjálfvirknivæðingu og sjálfvirkt gæðaeftirlit stofnunarinnar?
    e. Hversu oft koma út nýjar stórútgáfur (e. major version) af tækninni? Hversu langan líftíma hefur hver stórútgáfa? Er langtíma stuðningur fyrir útgáfur?
    f. Hversu vel fellur tæknin að forritunarumhverfum sem til eru í dag? Er tæknin studd í kóðaritlum og tólum? Hvernig er upplifun hugbúnaðarforritara?

3. **Mannafli** Hversu auðvelt verður að sækja nýja forritara til að viðhalda hugbúnaðinum í tæknistaflanum eftir 5 ár? 10 ár?
    a. Hversu auðvelt er að sækja forritara í dag? Eru margir á Íslandi með þekkingu á að reka og viðhalda lausnum í þessari tækni?
    b. Er tæknin aðgengileg óreyndari forriturum?
    c. Er auðvelt að sækja verktaka erlendis frá til að vinna að verkefninu?
    d. Hversu mikið hefur verið skrifað og er þekkt um lausnina? Eru upplýsingar á StackOverflow t.d.? Eru fáir sérfræðingar til? Er tæknin vel skjöluð?

Viðhaldsþolinn hugbúnaðararkitektúr

Við leggjum áherslu á að skipulag og uppbygging forritunareininga innan kerfisins styðji vel við það að upphaflegu hönnunarmynstri kerfisins sé viðhaldið rétt í lengri tíma og án aðkomu upphaflegu hugbúnaðarsmiðanna. Nauðsynlegt er að það sé auðvelt fyrir nýjan forritara, án sérstakrar handleiðslu annarra, að vinna í eldri virkni eða að útfæra nýja virkni á skjön við heildar högun lausnarinnar.

Arkitektúrinn þarf að aðgreina skýrt eftirfarandi þætti

- Móttöku og afhendingu gagna (e. request/response).
- Stýringar (e. controllers).
- Flæðistjórnun innan kerfisins.
- Gagnasókn og gagnaskrif.
- Ytri og innri gagnahögun (e. domains/models).

Til að tryggja þetta viljum við að allur hugbúnaður sem er skrifaður fylgi eftirfarandi högun og hugmyndafræði.
Það er okkar álit að þessi aðferðarfræði geti stutt best við þá langtímaáskoranir sem hugbúnaðurinn okkar mun standa frammi fyrir

1. SOLID principles + SoC (Separation of concerns)
2. ONION architecture + Principle of least knowledge / Multilayered architecture
3. CLEAN code
4. KISS (keep it simple silly)

5. DRY (don’t repeat yourself)
6. YAGNI (You aren’t going to need it)

Nauðsynlegt er að skipuleggja kóða á þann hátt að hægt sé að sannreyna á sjálfvirkan hátt (t.d. með einingaprófunum) að arkitektúr högun sé framfylgt rétt í lausninni. Við fjárfestum einnig í viðhaldi á sjálfvirkni til að reyna að tryggja að lausnir sem afhentar eru fylgi þessum högunum til lengri tíma.

Skýr og einföld sjálfvirknivæðing

Mikil áhersla er lögð á að nýta sjálfvirknivæðingu sem mest í hugbúnaðarumsýslu okkar. Snertir þetta alla hluta ferlisins og sérstaklega afurðaframleiðsluna og eftirlit.

Mögulegt er að draga mikið úr skjölun og villuhættu með því að útfæra prófanir í hugbúnaðarkóða sem framfylgja og styðja við eftirfarandi ferla.
Við gerum kröfu um að hugbúnaðurinn sem smíðaður er fyrir okkur uppfylli í það minnsta eftirfarandi sjálfvirkni

- Útgáfuferla: sjálfvirkar CI/CD pípur sem byggja og gefa út hugbúnaðinn, kubernetes samvirkni.
- Samþættingar: OpenAPI skilgreiningar.
- Prófanir: Sjálfvirkar eininga, viðmóts og samþættingaprófanir.
- Gæðaeftirlit: Sannreyningar á gæðum hugbúnaðar, skjölun sem styðja vottunarþarfir embættisins.
- Rekstur og eftirlit: Hugbúnaðarvirkni sem styður við sjálfvirka vöktun og bilanaleit í hugbúnaðinum.

Kjarnyrt skjölun

Markmiðið með skjölun er að þær upplýsingar sem þarf til að viðhalda langtímagæðum og högun lausna fylgi hugbúnaðarútgáfum fyrir okkur hverju sinni.

Góð og auðskilin skjölun eykur líftíma lausna til muna og felur í sér mikla langtíma hagræðingu fyrir nýja forritara og hönnuði sem koma að lausninni.

Markmið okkar í skjölun er ekki framleiðsla á þykkum doðröntum eða umritanir á almennri vitneskju, heldur kjarnyrt skráning á mannamáli sem lýsir högun og virkni viðkomandi lausnar, tengslum hennar við umheiminn og skráning á forsendum þeirra ákvarðana sem teknar voru.

Lögum og fyrirmælum um skráningu í sjúkraskrám framfylgt

Skráning og gagnavinnsla í heilbrigðisþjónustu fylgir lögum um sjúkraskrá og fyrirmælum um lágmarksskráningu í heilbrigðisþjónustu sem þarf að kynna sér áður en byrjað er að þróa nýjar hugbúnaðarlausnir fyrir embætti landlæknis. Upplýsingarnar er að finna á heimasíðu landlæknis.s

Höfum í huga að…

Enginn hugbúnaður er fullkominn. Hugbúnaði er ætlað að vera til staðar til að auðvelda fólki vinnu og aðgengi að upplýsingum. Hugbúnaðarsmiðir í samstarfi við sérfræðinga okkar þurfa að finna rétt jafnvægi milli þess að smíða og afhenda hugbúnaðarvörur á réttum tíma og innan kostnaðarmarka og hinnar fullkomnu lausnar. Markmið okkar verður ávallt að afhenda skjólstæðingum okkar bestu mögulegu lausnirnar innan tíma- og kostnaðarmarka.

Besti kóðinn og hugbúnaðarútfærsla er sú sem er auðskiljanleg, er auðveld í viðhaldi, styður vel við bilanaleit og stenst tímans tönn í rekstri. Þessari handbók er ætlað að styðja það að besta mögulega lausnin sé smíðuð í hvert skipti fyrir embættið.

Kóðageymslur

> 🌞 **Leiðarljós**
>
> Gæði og sjálfvirknivæðing er mikilvæg fyrir okkur, hins vegar viljum við ekki setja upp óþarfa þröskulda sem hindra geta vinnu verktaka meðan verkið er að vinnast. Verklaginu sem lýst er hérna er ætlað að reyna að ná fram sem mestum sveigjanleika og sjálfstæði allra aðila í samvinnu þegar unnið er að verkefnum fyrir okkur en á sama tíma tryggja endanleg gæði þess sem skilað er í kóðageymslur okkar.

Skipulag kóðageymsla (e. code repository)

Hugbúnaðarkóði fyrir aðgreindar þjónustur skal ekki vera geymdur saman í einni kóðageymslu (e. mono-repo fyrirkomulag). Ein kóðageymsla skal notuð fyrir hvern hluta lausnarinnar sem er gefinn út sjálfstætt.

Allar kóðageymslur okkar eru læstar og er ekki hægt að gera breytingar á innihaldi þeirra nema að fara í gegnum kóðarýni og samþykktarferli.

Einungis ein langlíf kóðagrein skal vera til í kóðageymslum landlæknis og skal hún nefnd main.

Sérfræðingar okkar stofna allar nýjar kóðageymslur í GitHub umhverfi MRH við upphaf verkefna.

Þegar send er inn beiðni þarf nýja kóðageymslan að innihalda nafn yfirverkefnis og svo nafn virkninnar sem búa á til. T.d. forritari er að vinna að lausn fyrir bókunarkerfi fyrir heilsuveru. Hann þarf kóðageymslu fyrir vefviðmót, vef-api og bakendaþjónustu, þá þyrfti að búa til þrjú repo í Github með nöfnunum:

- „heilsuvera-appointments-ui-web“
- „heilsuvera-appointments-api“
- „heilsuvera-appointments-service“

Nafnareglur á kóðageymslum

Kóðageymslur skulu fylgja eftirfarandi nafnareglum

1. Nöfn skulu ekki innihalda hástafi og nota bandstrik í stað bila eða undirstrika
    a. `heilsuvera-ui-web`
2. Allar kóðageymslur sem tilheyra sömu lausn skulu bera sama forskeytið. Í eftirfarandi dæmi er forskeytið „heilsuvera-“
    a. `heilsuvera-ui-web`
    b. `heilsuvera-api`
    c. `heilsuvera-service`
    d. `heilsuvera-database`
    e. `heilsuvera-ui-app-android`
    f. `heilsuvera-ui-app-ios`
    g. `heilsuvera-app-api`
3. Ekki skal nota tímabundin og bráðabirgða orð í nöfnum, líkt og -new, -old, -temp osfrv. Ekki skal heldur nota útgáfunúmer í nöfnum (t.d. -v1)

Nafna viðskeyti á kóðageymslur

Viðskeyti skulu reyna eftir bestu getu að vera eitt af eftirfarandi (eða samræmast einu af eftirfarandi). Athugið að þessi listi er ekki tæmandi, þau verkefni sem falla utan þessa skilgreinda ramma skulu gera sitt besta til að reyna að samræmast því mynstri sem lýst er í töflunni.

Hægt er að steypa saman viðskeytum ef þess er þörf, þá skal leitast við að samsetta viðskeytið sé uppbyggt frá hinu almenna að hinu sértæka (s.b. `-database-fhir` en ekki `-fhir-database`),

VIÐSKEYTI	LÝSING
-SERVICE	Fyrir bakendaþjónustur sem eru ekki aðgengilegar utan eldveggs
-API	Fyrir bakendaþjónustur sem eru aðgengilegar af internetinu (utan eldveggs)
-DATABASE	Gagnagrunnskóði
-DATABASE-APEX	Gagnagrunnskóði útfærður fyrir lausnir svipað og Oracle APEX.
-DATABASE-PIPELINE	Gagnagrunnskóði og annar kóði tengdur gagnameðhöndlunar pípum
-INFRA	Innviðakóði fyrir Kubernetes, Argo CD eða álíka (e. infrastructure as code)
-INFRA-ARGO-PPT	Innviðkóði ætlaður fyrir Argo CD á PreProduction umhverfi
-INFRA-K8S-PROD	Innviðakóði ætlaður fyrir Kubernetes á Production umhverfi
-PACKAGE-NUGET	Pakkar fyrir .NET Nuget kerfið
-PACKAGE-NPM	Pakkar fyrir Javascript NPM kerfið
-PACKAGE-MAVEN	Pakkar fyrir Java Maven kerfið
-JOBS	Forrit sem keyrð
-DOCS	Skjölun
-APP-WEB	Viðmótslausnir fyrir vef.
-APP-ADMIN-WEB	Viðmótslausnir fyrir stjórnenda vef
-APP-UI-ANDROID	Viðmótsforrit smíðað fyrir Android

-APP-UI-IOS	Viðmótsforrit smíðað fyrir Apple IOS
-APP-API	Bakendi sem einungis er ætlaður fyrir ákveðið viðmótsforrit. Ekki samnýttur með neinum öðrum forritum eða kerfum.
-APP-CLI	Console forrit, tekur við stýriskipunum úr console glugga en hefur ekki hefðbundið notendaviðmót.
-APP-DOCS-ANDROID	Skjölun fyrir viðmótsforrit smíðað fyrir Android
-APP-DOCS-IOS	Skjölun fyrir viðmótsforrit smíðað fyrir IOS
-AUTOMATION-ACTIONS	Github Actions sjálfvirknikóði skrifaður fyrir pípulínur
-AUTOMATION-PA	Sjálfvirknikóði skrifaður fyrir Microsoft Power Automate
-DASHBOARD-POWERBI	Mælaborð skrifað í Microsoft PowerBI
-DASHBOARD-WEB	Mælaborð skrifað sem vef sérsmíði
-CONFIG	Kóðageymsla sem inniheldur einungis stillingarskjöl (e. Configuration) enginn ótengdur forritunarkóði
-FHIR	Kóði sem tengist FHIR stöðun og útfærslum

Skilastjórar

Verktaki skipar einn eða fleiri aðila (skilastjóra) sem fá aðgang sem utanaðkomandi notandi (e. outside collaborator) og geta þá sótt afrit (e. fork) af kóða úr kóðageymslu MRH. Skilastjórar eru þeir einu sem geta skilað útgáfum af kóða inn í rýni til okkar.

Aðgangur skilastjóra sem ekki hafa skilað inn í kóðageymslur s.l. 3 mánuði verða sjálfvirkt fjarlægðir af kóðageymslum. Tilkynna ber til okkar ef verktaki skiptir um skilastjóra.

Hvernig ytri aðilar vinna með kóðageymslurnar

Við geymum allan kóða í **GitHub**, kóðageymslur nota **fork-and-branch** högun.

Við gerum kröfu um notkun á staðlaðri aðferðarfræði sem kallast **Fork-and-Branch**.

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo

Ferlinu er ætlað að gefa verktökum kost á að sinna vinnu við hugbúnaðinn í sínu umhverfi og samkvæmt sínum vinnureglum og öryggiskröfum. Við höfum svo einungis yfirsjón með endanlegum skilum á hugbúnaðarafurðum til útgáfu eða prófana. Verktakar og aðrir aðilar utan vébanda embættisins skulu ávallt vinna samkvæmt **Fork-and-Branch** ferlinu.

Skylda er að eyða **öllum** afritum (e. forks) úr tölvukerfum, afritum, skýjum og útstöðvum verktaka eftir að verki lýkur.

> Athugið að vegna takmarkana á samvirkni milli private repository forka og samvirkni milli Github Enterprise og annara útgáfa af Github þá er ekki hægt að notast við hefðbundna Fork-and-Branch högun. Þörf er á smávægis aðlögun á ferlinu tengdum skilum á kóða. **Verksali þarf að senda inn kóðagreinina til MRH og svo búa til PR á milli hennar og main.**

Fork-and-Branch ferlið

Athugið að nánari upplýsingar er að finna í Viðauka 3.

1. Verktaki býr til private `fork` af repoinu úr GitHub Landlæknis og vistar yfir á GitHub svæði verktaka.
    a. Ath, einungis þarf að búa til `fork` af `main` branchi (eða ákveðnu útgáfu tagi ef verkefnið kveður á um það).
    b. Ef ekki er hægt að búa til private `fork` af MRH repoinu (t.d. ef verksali er í Github Enterprise) þá nægir að viðhalda bara tengingu við `main` branch MRH (mögulegt með `clone` og `upstream` stillingum).
2. Búa til `git clone` af forkinum þínum á local vélina þína
3. Bæta við nýjum remote sem heitir `upstream` fyrir þitt klón sem bendir á GitHub repo landlæknis
4. Byrja að vinna að verkinu og committa breytingar í þinn fork
    a. Mælt er með að notist við feature branches.
5. Þegar kemur að gefa út útgáfu af lausninni, þá búa til útgáfu branch byggða á forkinum þínum
    a. Athugið nafnareglur á útgáfu branches.
6. `commit` þær breytingar sem þarf fyrir útgáfu (uppfæra pakka, númer etc) í útgáfu branchið
7. `push` útgáfu branchinu upp **í GitHub MRH repoið**
8. Opna PR frá útgáfubranchinu sem þú varst að senda til MRH yfir í main branch hjá MRH.
    a. Athugið að ekki er gerð krafa um að commit séu `squashed`.
9. PR verður code reviewað og leiðréttinga gæti verið þörf, leiðréttingar gerast í útgáfu branchið.
    a. Eftir að PR er samþykkt þá verður búið til tag í GitHub landlæknis fyrir útgáfunúmerið
10. Eftir að PR hefur verið samþykkt, tögguð og sameinuð við Repoið hjá Landlækni, þá þarftu að uppfæra forkinn í þínu GitHub
    a. `git pull upstream main`
    b. `git push origin main`

Við hvetjum verktakar til að íhuga að sjálfvirknivæða þennan feril í einhverri mynd sín megin til að forðast óþarfa tafir sem geta myndast ef skil eru framkvæmt handvirkt af ákveðnum aðila (t.d. gæti verið hagkvæmt að sjálfvirknivæða skref 5 til 8). Hægt er að óska eftir því að fá Access Token frá MRH (eða skilastjórar geta útbúið sinn eigin personal access token) til að nota í útfærslu á sjálfvirknivæðingu.

Nafnareglur á kóðagreinum

Kóðageymslur okkar taka einungis við breytingarbeiðnum (Pull requests, eða PR) sem fylgja eftirfarandi nafnareglum.

Titlar/nöfn á breytingabeiðnum frá skilastjóra inn í kóðageymslur okkar þurfa að vera eins og nafn á viðkomandi kóðagrein*. Þ.e.a.s. ef Git greinin heitir „qa/2025.01.01-rc1“ þá þarf PR beiðnin sem send er upstream einnig að hafa titilinn „qa/2025.01.01-rc1“.

Kóðagrein (e. branch name)	**Lýsing**
release/2024.08.05 release/2024.08.05-hotfix1	Loka útgáfur af fyrirfram ákveðnum útgáfum eða neyðarútgáfum af hugbúnaði sem ætlaður er til útgáfu á pre-production og production umhverfum. Athugið einungis útgáfur merktar á þennan hátt munu vera leyfðar á production umhverfi. Athugið að útgáfunúmer mega ekki enda á „-rc1“ í production.
qa/2024.08.05-rc1	Útgáfur eingöngu ætlaðar til sjálfvirkrar útgáfu á pre-production umhverfi. Ath: viðskeyti (í þessu tilviki „-rc“) verður að fylgja. Viðskeytið má vera hvaða texti sem er en verður að enda á -rcNN, þar sem NN er teljari (sbr -rc1, -rc10).
test/2024.08.05-dev	Prófunarútgáfur af hugbúnaði sem ætlaður er til sjálfvirkrar útgáfu á test umhverfi.

*Helsta ástæðan fyrir því af hverju PR og kóðagreinar þurfa að heita það sama er til að geta tryggt skýran rekjanleika milli merkinga í kóðagreinum, breytingabeiðna, merkinga í prófanaskýrslum og veittra samþykkta í breytingabeiðnum til útgáfu.

Gæða og útgáfuferlar

Sjálfvirknivæðingarmarkmið

> 🌞 **Leiðarljós**
>
> Við leggjum mikla áherslu á réttleika og gæði hugbúnaðarins sem smíðaður er í nafnið þess. Til að tryggja einsleita og hlutlæga nálgun á gæðaeftirliti þá innihalda kóðageymslur landlæknis sjálfvirka ferla til að framfylgja eftirlitinu. Skilvirkni sjálfvirknivæðingarinnar byggir á að þeir sem vinna í kóðageymslunum tileinki sér ákveðnar nafnahefðir og vinnureglur.

Ábyrgðir og væntingar

Umhverfunum sem lýst er í þessum kafla eru þau sem embættið hýsir sjálft og býður verksölum upp á möguleikann á að gefa út á. Ekki er ætlast til þess að verksali útfæri neitt af þeim umhverfum sem lýst er hér í sínum umhverfum eða þróunarumhverfum. Vegna flækjustigs umhverfa okkar þá eru ephemeral umhverfi ekki möguleg.

Gæðaferlar og eftirlit sem lýst er hér er einnig útfært og hýst af embættinu og keyrist sjálfvirkt við skil. Ferlunum og kröfunum er lýst hér svo verksali geti gert ráðstafanir við daglega þróunarvinnu til að tryggja að samsvarandi gæðaeftirliti sé sinnt tímanlega til að forðast óþarfa umfram vinnu við verkskil.

Tæknifólk okkar leggur til grunnvirkni þeirrar sjálfvirknivæðingar sem keyrast skal í kóðageymslum MRH við skil og vinnur, við skil eða fyrr, í samstarfi við verksala hugbúnaðinum að aðlaga grunnvirknina að þörfum skilaafurðanna.

Verksali ber sjálfur ábyrgð á að útfæra alla þá sjálfvirkni sem þeir telja æskilega til að sinna daglegum störfum á sínum vinnustöðvum, þróunarumhverfum og kóðageymslum. Einnig ber verksali ábyrgð á að

setja upp og viðhalda þeim þróunarumhverfum og sýndarþjónustum (e. mocked services) sem nauðsynlegt er til að sinna daglegum þróunartengdum störfum hjá sínu starfsfólki.

Skipulag útgáfunúmera

Hugbúnaðurinn okkar skal notast við dagsetningarútgáfur (e. Calendar Versioning, http://Calver.org). Dagsetningarútgáfur gefa breiðari hópi notanda betri yfirsýn yfir innihald útgáfa, samvirkni milli útgáfa og hversu nýlegur hugbúnaðurinn er. Semantic versioning (þ.e. 1.0.0) á ekki að vera notað.

Númerin skulu fylgja eftirfarandi formi

YYYY.0M.0D-*breyta*

- YYYY - Fjögurra stafa ár - 2024, 2026, 2106…
- 0M - Tveggja stafa mánaðarnúmer, núll fremst ef þarf - 01, 02 ... 11, 12
- 0D - Tveggja stafa mánaðardagur, núll fremst ef þarf - 01, 02 ... 30, 31
- breyta - Valkvæður viðbótar texti fyrir séraðstæður, t.d. "dev", "alpha", "beta", "rc1", “hotfix”, o.s.frv.. Ef gefa þarf út margar breytu útgáfur innan sama dags skal nota “-breytaN” þar sem N - er tala 1…99)

Undantekning frá þessari reglu er hugbúnaður sem gefin eru út í app-verslunum á borð við Google store og App store, þar sem gerð er krafa um að nota Semantic versioning fyrir ákveðin útgáfunúmer.

Allur hugbúnaður hjá okkur hlýtur útgáfunúmer á sjálfvirkan hátt eftir að gæðastýringarferlum er lokið. Útgáfunúmer ákvarðast af nafni á þeirri kóðagrein sem er rýnd hverju sinni.

Sjálfvirkir gæðastýringarferlar

Athugið að öllum sjálfvirkum ferlum sem lýst er hér eru stilltir og keyrðir af GitHub Actions hjá Landlækni við skil. Ekki er krafa um uppsetningu á þessum sömu ferlum hjá verktökum en við hvetjum alla til að hafa þá til hliðsjónar við hönnun á píplínum sem keyra verktaka megin. Nánari lýsingar á þessum ferlunum er að finna í Viðauka 1.

Lausnum er skilað til okkar sem PR (e. pull request). Fyrir hver kóðaskil sem eru búin til í kóðasöfnum okkar fer eftirfarandi ferill í gang

1. Kóðinn er prófaður m.t.t. uppbyggingar lausnar
2. Kóðinn fer í gegnum build ferli
3. Kóðinn fer í gegnum einingaprófunarferli
4. Kóðinn fer í gegnum sjálfvirkt kóðaryni og kóðagæðaeftirlit
5. Keyrslustýringar (docker) fer í gegnum öryggisprófun
6. Allur gagnagrunnskóði fer í gegnum gæðaeftirlit

Skili eitthvert af þessum sjálfvirku skrefum villu er PR sjálfkrafa hafnað og höfundi ber að gera viðeigandi breytingar til að tryggja að gæðakröfur séu uppfylltar.

Hálf-sjálfvirkir gæðastýringarferlar

Ef að sjálfvirkir ferlar tilkynna engar villur þá fer í gang kóðarýni á skilunum. Þessi kóðarýni eru gerð af sérfræðingum okkar við skil á lausnarkóða. Þróunarhandbók liggur til grundvallar við kóðarýni.

Sérstök áhersla er lögð á eftirfarandi atriði í kóðarýnum.

1. Réttleiki hugbúnaðar og gagnagrunnskóða
2. Langtíma heilsa og viðhalds eiginleikar lausnar
3. Arkitektúr lausnarinnar
4. Gæði skjölunar
5. Uppbygging og skipulag skráa og lausnar
6. Gæði og viðhalds eiginleikar sjálfvirkra prófana
7. Öryggis og áhættumat á ytri tengslum (e. dependency) við aðra hugbúnaðarpakka og lausnir.

Ef kóðarýnir (e. reviewer) metur að eitthvað af áhersluatriðum sé ábótavant er PR hafnað og þess óskað að höfundur geri breytingar til að tryggja að gæðakröfur séu uppfylltar. Einungis eru rýnd PR sem eru hluti af qa og release skilum. Sjá nánari leiðbeiningar fyrir kóðarýnendur í viðauka.

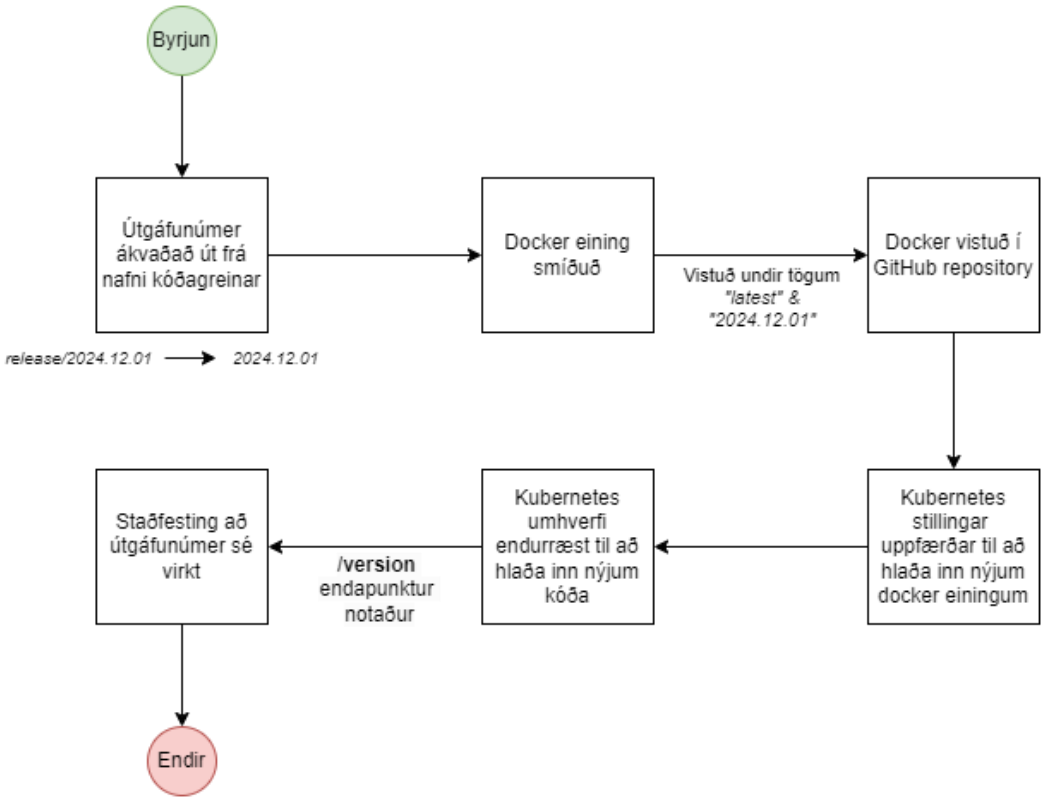
Sjálfvirkir útgáfuferlar

Allur hugbúnaður sem ætlaður er til útgáfu á rekstrarumhverfum okkar er útbúinn til útgáfu með sjálfvirkum ferlum. Þetta á við um öll umhverfi, test, pre-production og production. Þessi sjálfvirknivæðing er til að tryggja gæði útgáfuferla og til þess að draga úr líkum á mannlegum mistökum við útgáfu hugbúnaðar.

Sjálfvirk útgáfuferli fylgja eftirfarandi skrefum

1. Kóði fær sjálfvirkt útgáfunúmer (e. tag) í Git kerfinu
2. Smíði Docker einingar (e. container)
3. Docker einingu er gefið sama útgáfunúmer og vistuð í miðlæga geymslu (e. container registry) í GitHub embættisins
4. Kubernetes stillingar eru uppfærðar með nýjum útgáfunúmerum og stillingum
5. Nýjar docker einingar eru virkjaðar á Kubernetes umhverfi og lausnin er gefin út

Útgáfuferillinn í mynd



Á hvaða umhverfi eru útgáfur leyfðar á?

Sjálfvirkir útgáfuferlar bjóða upp á að gefa út útgáfur á eftirfarandi umhverfi eftir því hvernig kóðagrein var verið að senda inn.

Umhverfi/Kóðagrein	**Sjálfgefin útgáfutög**	**release/YYYY.MM.DD**	**qa/YYYY.MM.DD**	**test/YYYY.MM.DD**
Production	*Nýjasta CalVer*	**Sjálfvirkt/Handvirkt**	**Ekki leyft**	**Ekki leyft**
Pre-Production	*Nýjasta CalVer-rcNN*	**Sjálfvirkt**	**Sjálfvirkt**	**Ekki leyft**
Test	*latest-test*	**Handvirkt**	**Handvirkt**	**Sjálfvirkt**

Sjálfvirkar gæða og öryggisprófanir

Gæðaprófanir

Allar PR sem eru búnar til í kóðageymslum okkar fara í gegnum sjálfvirkar gæða og öryggisprófanir, þessar prófanir þurfa að standast kröfur til þess að kóðinn sé samþykktur og að hægt sé að gefa hann út á rekstrarumhverfum.

Stillingarnar á gæða og öryggisprófunum er í höndum embættisins og er óleyfilegt að breyta þeim án samþykkis verkefnastjóra okkar.

Við stólum helst á SonarCloud til að framkvæma sjálfvirkar öryggis og gæðaprófanir á kóða.

Athugið, SonarCloud prófanir eru stilltar þannig að viðvaranir eru meðhöndlaðar sem villur (e. treat warnings as errors). Lagt er til að forritarar stilli þróunarumhverfi sín á sama hátt.

Gerðar eru prófanir á stillingarskrám fyrir þýðendur og þarf að vera kveikt á eftirfarandi stillingum

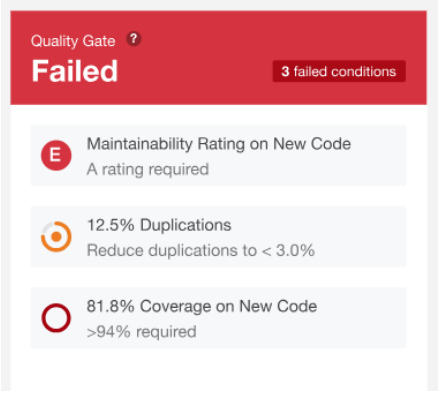
- C# .csproj skrár (eða samsvarandi skrár, sbr. Directory.Build.props)
    - CodeAnalysisRuleSet, GenerateDocumentationFile, tög séu til staðar
    - TreatWarningsAsErrors sé til staðar og sett á „true“
    - Að eftirfarandi pakkar séu til staðar
        - StyleCop.Analyzers
        - StyleCop.CSharp.Async.Rules
        - Microsoft.CodeAnalysis.NetAnalyzers
        - StyleCop.CSharp.Async.Rules
- Typescript (tsconfig.json)
    - Eftirfarandi stillingar þurfa að vera settar á „true“
        - strict, alwaysStrict, noImplicitAny, strictNullChecks, strictFunctionTypes, exactOptionalPropertyTypes, noImplicitReturns, noUnusedLocals, noUnusedParameters, useUnknownInCatchVariables, sourceMap, noUncheckedSideEffectImports
    - Ath, ekki slökkva á undirstillingum sem heyra undir yfirstillingar á borð við „strict“
        - T.d. ekki setja strict=true og svo strictFunctionTypes=false
    - Ef **allowJS** er set á „true“, þá verður **checkJs** að vera sett á „true“ líka

Sjálfvirk gæðahlið (e. Quality Gates)

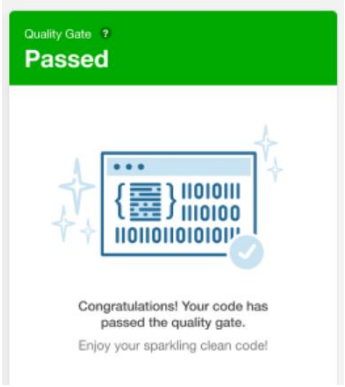
SonarCloud umhverfið okkar hefur staðlaðar stillingar á gæðahliðum fyrir lausnir. Almennt séð ef gæðahlið hafnar breytingum (e. failed quality gate) þá uppfyllir kóðinn ekki lágmarks gæðakröfur og skilum er hafnað.

Gæðahlið eru stillt þannig að þær bera kóðann í skilunum saman við núverandi stöðu í kóðagrunninum. Á þann hátt ætti að vera tryggt að gæði kóðans batni án þess að leggja á forritara möguleg fyrirliggjandi vandamál í eldri kóðahlutum sem eru ótengd þeim breytingum sem verið var að gera (e. clean-as-you-code aðferðarfræði).

Sjá nánari lýsingu á gæðakröfum í kafla um prófanir.



Gæðahlið sem hefur hafnað kóðaskilum



Gæðahlið sem samþykkti kóðaskil

Docker öryggisprófanir

Allar lausnir þurfa að innihalda docker skipanir og þurfa þær skrár að standast öryggisprófanir (e. container scanning). PR er hafnað ef athugasemdir finnast með hæsta viðvörunarstig (e. “Critical”). Eftirfarandi tegundir prófana eru alltaf gerðar á docker einingum (viðbótar prófanir gætu verið keyrðar í ákveðnum kringumstæðum).

1. Greining á grunn docker einingum sem notaðir eru (base image scanning)
2. Greining á stýrikerfispökkum sem grunneiningar stóla á (third-party dependencies)

Skil á hugbúnaðarafurðum

Við hvetjum eindregið til þess að vinnuhópar vinni samkvæmt Minimum-Viable-Product (MVP) högun og að skilastjórar skili reglulega hugbúnaði og hugbúnaðarafurðum inn í rýni til okkar (að lágmarki einu sinni á 3 mánaða fresti). Með því er hægt að leysa fyrr úr mögulegum málum og gera stefnuleiðréttingar ef þörf krefur.

Mælst er til að reynt sé að skila reglulega inn útgáfum af þeirri virkni sem kláruð er til rýni og samþættingaprófana.

Hverju á að skila?

Vinnuhópar eru hvattir til að skila reglulega inn QA útgáfu af öllu því sem þeir hafa hugsað sér að skila í lok verkefnisins til að hægt sé að rýna verkið eins og það vinnst. Skil ná til allra þátta verkefnisins, þ.m.t. kóða, prófana og skjölunar.

Hver skilar?

Skilastjóri hvers verkefnis skilar vörum inn til okkar. Hægt er að skilgreina einn eða fleiri skilastjóra ef þörf krefur.

Hvenær er æskilegt að skila inn?

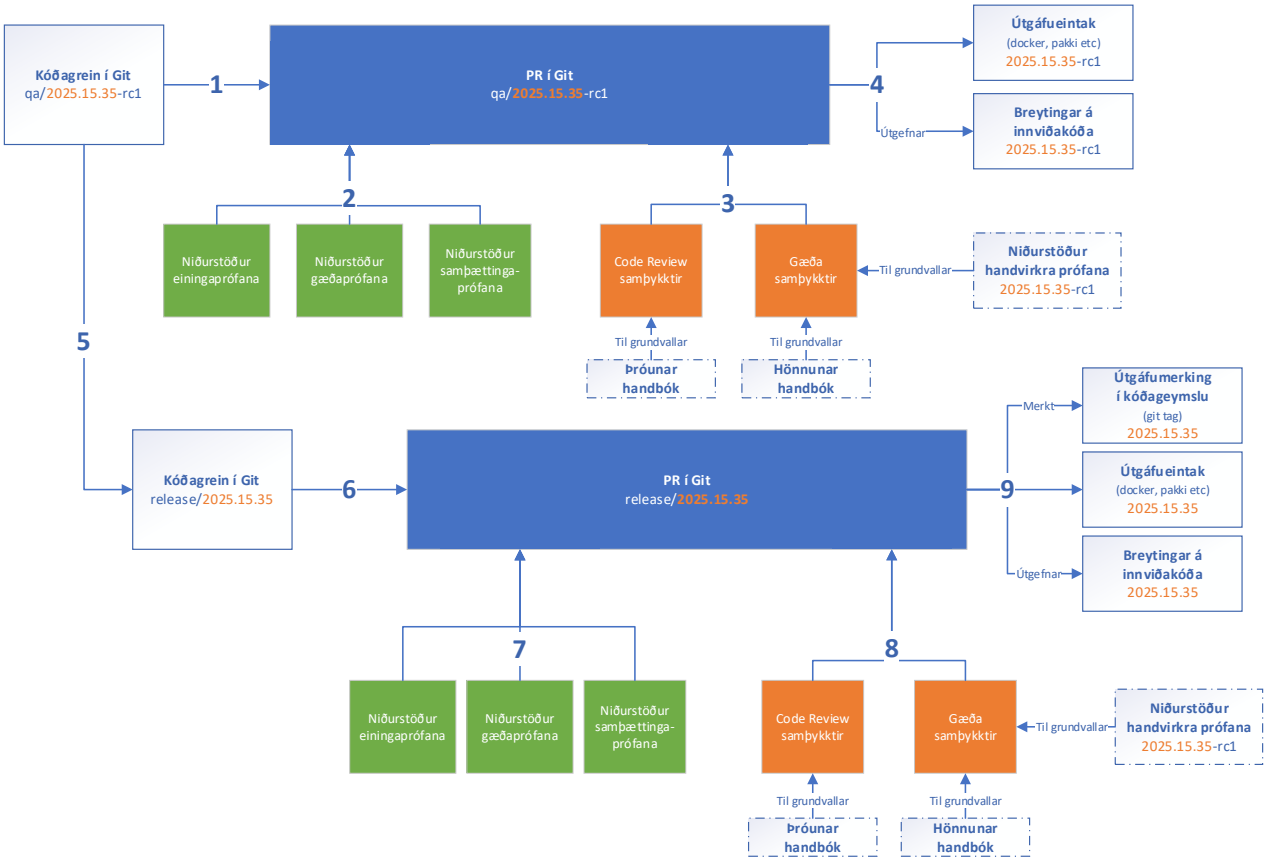
Skil	1x í viku	>=1x í mánuði	<1x í mánuði	Í lok verkefnis
Á test umhverfi	Eftir þörfum (ekki þó oftar en 1x á dag)			
Á pre-production umhverfi	Í lagi	Æskilegt	Æskilegt	Óæskilegt
Á production umhverfi	Óæskilegt	Óæskilegt	Í lagi	Æskilegt

Rekjanleikakröfur í skilum og gæðaprófunum

Nauðsynlegt er að halda rekjanleika á sem einfaldastan hátt milli kóða sem skilað er inn, gæða og prófanaskýrsla og samþykkta fyrir hver skil.

Rekjanleika er náð fram með því að

- Samræma nöfn á milli kóðagreina, breytingabeiðna og handvirkra prófana (1,5,6)
- Breytingabeiðnir (PR) innihaldi niðurstöður úr prófunum (gæða-, eininga- og samþættingaprófanir) (2, 7)
- Breytingabeiðnir (PR) innihaldi samþykktir á kóðaskilum og gæðamati (3, 8)
- Breytingabeiðnir (PR) innihaldi yfirlit (log) yfir hvernig hugbúnaður var byggður og prófaður
- Samræma nöfn milli QA og RELEASE kóðagreina (5)


Kóðagrein í Git
qa/2025.15.35-rc1
1
PR í Git
qa/2025.15.35-rc1
4
Útgáfueintak
(docker, pakki etc)
2025.15.35-rc1
Útgefnar
Breytingar á innviðakóða
2025.15.35-rc1
2
3
Niðurstöður einingaprófana
Niðurstöður gæðaprófana
Niðurstöður samþættinga-prófana
Code Review samþykktir
Gæða samþykktir
Til grundvallar
Niðurstöður handvirkra prófana
2025.15.35-rc1
Til grundvallar
Til grundvallar
Þróunar handbók
Hönnunar handbók
5
Kóðagrein í Git
release/2025.15.35
6
PR í Git
release/2025.15.35
9
Merkt
Útgáfumerking í kóðageymslu
(git tag)
2025.15.35
Útgáfueintak
(docker, pakki etc)
2025.15.35
Útgefnar
Breytingar á innviðakóða
2025.15.35
7
8
Niðurstöður einingaprófana
Niðurstöður gæðaprófana
Niðurstöður samþættinga-prófana
Code Review samþykktir
Gæða samþykktir
Til grundvallar
Niðurstöður handvirkra prófana
2025.15.35-rc1
Til grundvallar
Til grundvallar
Þróunar handbók
Hönnunar handbók

Forritunarkóði

Uppbygging og hönnun

> 🌞 **Leiðarljós**
>
> Við gerum ráð fyrir 30+ ára rekstrar- og viðhaldstímabils fyrir hugbúnaðarlausnir. Af þessum sökum er mikilvægt fyrir okkur að velja bæði arkitektúr, högun, forritunarmál og forritunaraðferðir sem styðja vel við slík langtíma verkefni. Sérstaklega þarf að hafa í huga að margir og mismunandi hugbúnaðarsmiðir og hönnuðir geta komið að verkefnum í styttri eða lengri tíma. Tæknival og forritunarstíll þarf því að vera óháð tímabundnum vinsældum í straumum og stefnum. Tækni sem valin er þarf að vera líkleg til að laða að sér nýtt hugbúnaðarfólk til lengri tíma.

Forritunarmál og framework

Bakendakerfi / API

Bakendakerfi okkar eru flest skrifuð í .NET og C#, því hefur verið ákveðið til að einfalda tækniumhverfi að bakendakerfi, s.s. APIs og þjónustur sem eru hluti af keyrsluumhverfi Landlæknis, skulu öll skrifuð í C# og notast við .NET umhverfið. Hinsvegar ef brýn nauðsyn eða sértækar tæknilegar ástæður krefjast þess að annað forritunarmál og umhverfi sé valið er leyfilegt að velja annað umhverfi gegn þess að fá leyfi sérfræðinga okkar og tryggja að sjálfvirkir gæðaferlar séu útfærðir fyrir viðkomandi tæknistakk.

Ávallt skal notast við nýjustu LTS (e. long term support) útgáfuna af þróunar og keyrsluumhverfum, þ.e.a.s. útgáfan sem verður í gildi við áætlaðan útgáfudag hugbúnaðarins.

Bakendalausnir skulu leitast til við að nýta til hins ýtrasta innbyggða virkni í þau forritunar framework sem þau eru hönnuð fyrir, sbr ASP.NET.

Framendar / Web

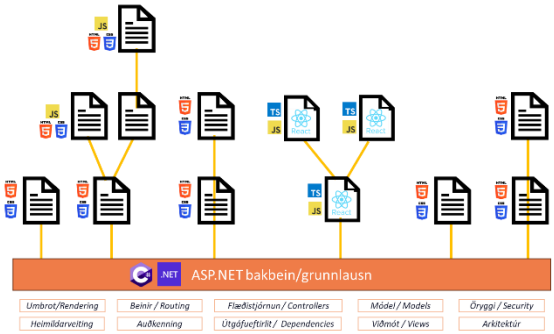
Forritunarlausnir fyrir framenda skulu leitast við að nýta á sem bestan máta innbyggða staðla til vefsíðubirtinga s.s. HTML5 og CSS. Hafa skal það að sjónarmiði að að það er líklega einfaldast fyrir forritara að koma að 20 ára gamalli lausn til að gera viðbætur eða breytingar ef lausnin er að mestu leyti uppbyggð sem stöðluð HTML lausn með einfaldan tæknistafla.

Notast skal við þau framenda framework sem boðið er upp á sem hluta af ASP.NET og hönnunarmynstur sem eru ráðandi í þeim lausnum sem þegar eru til staðar (t.a.m. MVC eða Razor, WebForms eru ekki leyfileg). Nota skal C# forritunarmálið. Íhuga skal notkun HTMX fyrir gagnvirkni í framenda og nýta skal web components aðferðarfræði þar sem mögulegt er.

Allir framendar sem eru smíðaðir, verða að fylgja útgefnum hönnunarstöðlum og útlitsstílsniði okkar

- Design system – Desktop UI Library
- @storybook/cli - Storybook

Til útfærslu á gagnvirkni skal reynt að nýta hugbúnaðartækni sem útbýr og þjónustar vefsíðuefni af netþjóni (e. server-side rendering eða SSR) fram yfir virkni sem keyrir eingöngu í vafra notanda. Dæmi um slíka tækni er MVC og Razor í ASP.NET. Upplýsingakerfi sem stóla á aðgengi leitarvéla að vefslóðum (e. URLs), kerfi sem krefjast ekki sérstakrar gagnvirkni í viðmóti, eða þurfa ekki að styðja slitrót samband við alnetið skal útfæra alfarið sem SSR kerfi. Ekki skal nota Javascript SPA framework eins og React, Angular, Vue o.þ.h. nema eðli verkefnisins krefjist þess, að ekki sé möguleiki að útfæra viðkomandi virkni með .NET tækni og að leyfi til þess hafi verið fengið hjá okkur. Þróunaraðilar eru hvattir til að kynna sér HTMX aðferðarfræði.



Ef SSR nálgun er ekki fýsileg og leyfi hefur verið fengið til að útfæra SPA vefsíðu sem keyrir í vafra notenda þá skal samt sem áður að nota ASP.NET sem bakbeinið í lausninni ef mögulegt er. Stuðst skal við Javascript forritunarmálið á þann hátt að hægt sé að gæta að kóðagæðum (e. type checking og static analysis) í forritunarkóða til að lágmarka hættu á forritunarmistökum. Ef Javascript útgáfan styður ekki slíka virkni skal notast við nýjustu LTS útgáfu af TypeScript málinu. Einungis skal notast við React við gerð SPA veflausna.

Ekki skal notast við SSR eða SSG (e. server-side generation) virkni sem útfærð er fyrir SPA (Single Page Application) forritunarsöfn, dæmi um SSR söfn sem ekki eru leyfileg er t.d. Next.js og önnur sambærileg sem byggja á BBF (e. backend for frontend) högun.

Ekki skal smíða lausn í forritunarmáli eða umhverfi sem ekki hefur verið notað áður nema með leyfi sérfræðinga okkar.

Ekki skal undir neinum kringumstæðum útfæra veflausnir fyrir okkur í vefumsjónarkerfum (e. web content management systems) án skriflegs samþykkis frá bæði tæknilegum sérfræðingum og verkefnastjórn okkar.

Snjallforrit / Mobile

Almenn regla er að snjallforrit ætluð fyrir snjalltæki (Android eða iOS) skulu í lengstu lög vera hönnuð og útfærð sem snjalltækjavænar vefsíður frekar en sem snjallforrit. Þumalputtareglan er að útfæra skuli hefðbundna snjalltækjavæna veflausn ef lausnin krefst ekki mjög sértækrar tækjavirkni (sem ómögulegt eða ópraktískt er að gera í veflausn).

Í þeim undantekningartilvikum þar sem ekki er hægt að útfæra snjalltækjavænar veflausnir þá skal leitað samþykkis sérfræðinga okkar áður en ákveðið forritunarumhverfi eða framework er valið.

Öll viðmót fyrir snjallforrit skulu vera hönnuð og virkni þeirra aðgengileg á ásættanlegan máta á a.m.k. 75% af þeim snjalltækjum og útgáfum þeirra sem eru í notkun af skilgreindum markhópi verkefnisins á útgáfudegi þess.

Hönnun og arkitektúr

Allur arkitektúr og hönnun þarf að vera unnin í samráði við sérfræðinga okkar. Hugbúnaðarsmíði skal ekki hefjast fyrr en samþykki á hönnun og arkitektúr liggur fyrir. Hönnun á högun og venslum gagna skal einnig liggja fyrir áður en hugbúnaðarsmíði hefst.

Lausnarbanki

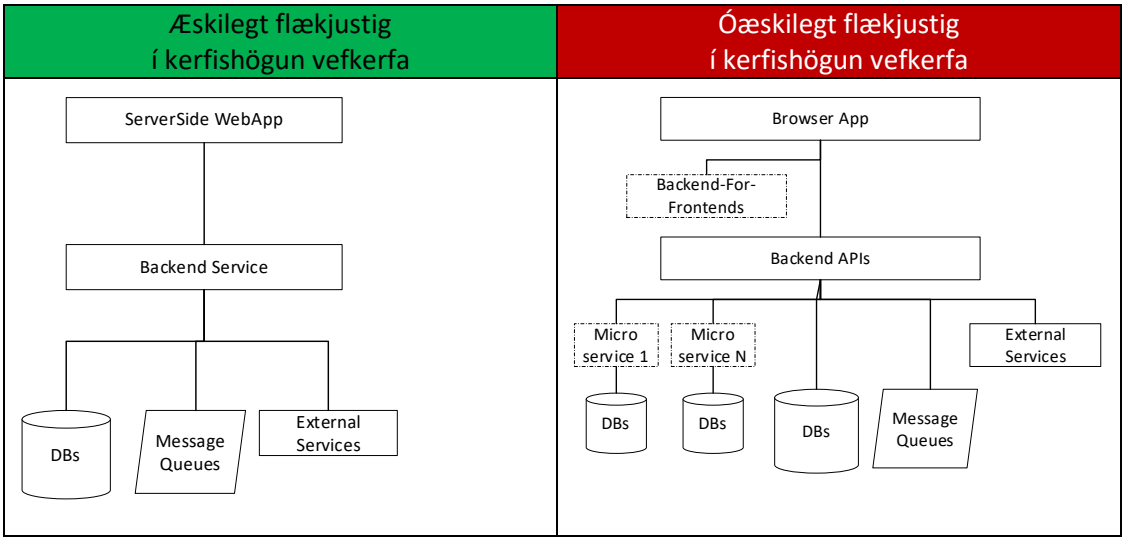
Við höldum úti lausnarbanka sem inniheldur lausnir sem lýsa einni mögulegri útfærslu á þeim reglum og viðmiðum sem þessi handbók lýsir. Aðgengilegur á https://github.com/MRH-Landlaeknir-Samples. Hægt er að fá aðgang að bankanum með því að óska eftir aðgangi frá tækniteymi landlæknis eða á mrh@landlaeknir.is. Athugið að embættið tekur enga ábyrgð á mögulegum villum sem kunna að vera í lausnum í lausnarbanka sínum og er ekki ætlast til að kóðinn þar sé notaður óbreyttur eða órýndur í lausnir verksala fyrir embættið (sjá fyrirvara).

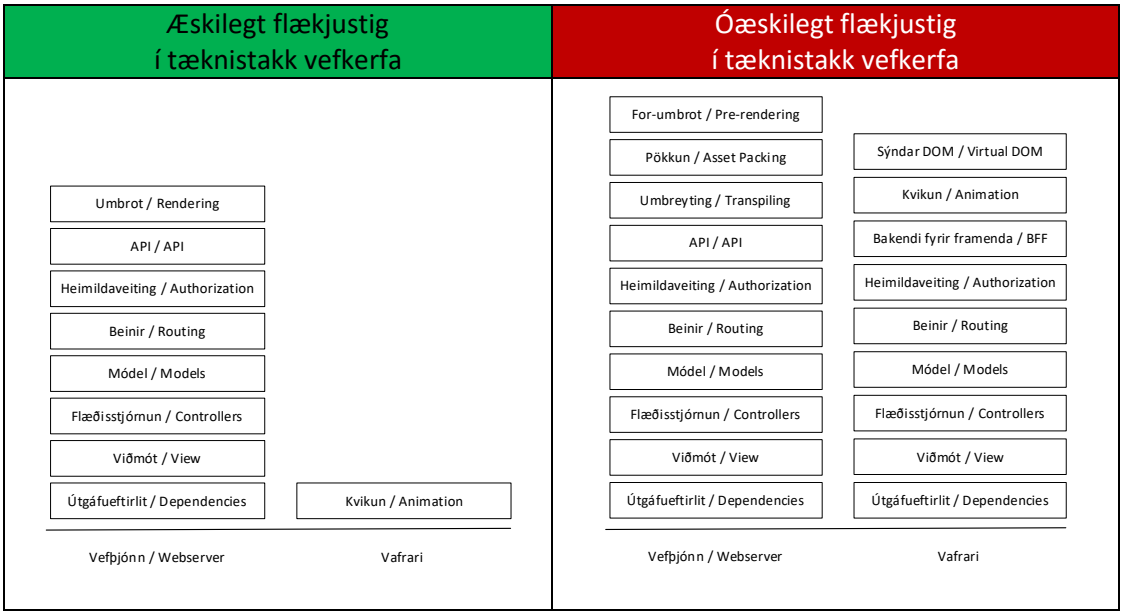
Viðmiðunararkitektúr

Kerfi smíðuð fyrir okkur skulu styðja samvirkni við HL7 staðalinn í heilbrigðisgögnum og samþættingu við FHIR kerfiseiningar landlæknis. Kerfi skulu reyna eftir fremsta megin að lesa, meðhöndla og skrifa gögn samkvæmt HL7 staðli eins og hann er skilgreindur á landsvísu og nota miðlægar FHIR innviðalausnir landlæknis.

Kerfi skulu útfæra róttæklega einfaldan tækniarkitektúr, sem tekur mið af að fækka einingum í hugbúnaði til að stuðla að langtíma stöðugleika án þess þó að fórna langtíma viðhalds- og viðbótareiginleikum.

Til hliðsjónar þá skal miða við að útfærsla tæknilausnar aðhyllist eftirfarandi arkitektúr og tæknistakk. Þetta á við um allar síðu-miðaðar (e. page centric) lausnir sem krefjast ekki sérstakrar gagnvirkni[1] eða er ekki ætlað að hafa sér-skipað langtíma viðhaldsteymi.





[1] Dæmi um sérstaka gagnvirkni

- Drag-and-drop og sérhæfður innsláttur, s.s. töflureiknar, grafískhönnun, textaritlar, spjallforrit.
- Hugbúnaðarlausnin þarf að virka ótengd (e. offline) eða í mjög slæmum tengiskilyrðum.
- Gagnvirk kvikun/virkni (e. animation), s.s. kortabirting, leikir, myndbönd, myndræn gagnaframsetning.

Ástæður fyrir rótækilega einfölduðum tækniarkitektúr

Er rótæk einföldun þá það sama og „að velja leiðinlega tækni“ (e. choose boring technology)? Já og nei, báðar stefnur eru líkar, þ.e.a.s. að lágmarka áhættu og auka skilvirkni. Og þó að með því að velja leiðinlega tækni er verið að lágmarka fjölda af tæknieiningum þá einblínir sú stefna á að velja sannreynda og áreiðanlega tækni sem hefur vel þekkt lausnarmengi. Það er hinsvegar enn möguleiki að byggja flóknar lausnir með leiðinlegri tækni. Rótæk einföldun gengur út á að hafa eins fáar einingar og hluti á hreyfingu eins og mögulegt er. Endurnýting á tækni sem þegar er hluti af tæknistakk umfram það að draga inn nýja tækni.

Einfaldur tæknistakkur gerir hugbúnaðarfólki færi á að öðlast dýpri tækniþekkingu og einfaldar lærdómskúrfu nýs fólks. Með færri hlutum þá drögum við úr líkum á að eitthvað brotni og gerum stillingar einfaldari.

Ástæðan fyrir því af hverju Landlæknir aðhyllist slíka stefnu er að ekki er til staðar fjármagn eða burðir til að standa undir þeim viðhalds- og uppfærslukostnaði sem lausnir líkt og gagnvirk Javascript viðmót í vafrara krefjast. Við þurfum einnig að stýra og takmarka fjárfestingarbindingar tengdar tíðum breytingum, endursmíðum og endurnýjun á forritasöfnum notuðum í þegar afhendum hugbúnaði sem þegar er í rekstri.

Æskilegir tæknistaflar

Framendi (vef)	**Bakendi**	**Gagnagrunnur**
ASP.NET / C# HTMX, HTML, CSS Javascript (að mjög litlu leyti) Docker	ASP.NET / C# JSON SQL + Dapper.NET (ORM) Docker	FHIR / HL7 MS-SQL
Rekstrarumhverfi	**Kóðageymslur og útgáfa**	**Auðkenning/Heimildir**
Kubernetes + Argo CD Structured logging (Coralogix) Prometheus (Coralogix)	Github + Github Actions SonarCloud + Sonarlint + StyleCop	OAuth2 + JWT eða ApiKeys Island.is eða Saga Tokens (ef við á) Claims heimildir

Þolgæði lausna og samskipta

Lausnir og samskipti þeirra við aðrar þjónustur skulu vera hannaðar með þolgæða sjónarmiði (e. fault tolerance). Það þýðir að allar lausnir skulu meðhöndla á viðeigandi og ásættanlegan hátt stuttar eða viðvarandi truflanir í samskiptum, tímabundum niðritíma tengdra þjónusta og gagnagrunna, óvæntum breytingum á samskipta sniði (e. data schema) og önnur álíka tilvik. Upplifun notenda skal hafa í fyrirrúmi þegar truflanir í samskiptum eru meðhöndlaðar og tilkynntar.

Leitast skal við að nýta forritunarsöfn til einföldunar á samskiptum við ytri API þjónustur og gagnagrunna og til að sjá um að útfæra þolgæðavirkni (t.d. retry og circuit-breaking).

Í þolgæðum þá hafa í huga að nota eftirfarandi mynstur

- Endurtekningar (e. retry). Útfæra skal lágmarks endurtekningar á fyrirspurnum. Endurtekningar skulu nýta sér bæði jittering og staggered backoff periods.
- Ekki skal útfæra sérsmíðaðan þolgæðakóða, forritarar skulu nota viðurkennd forritunarsöfn, sbr
- .NET: Polly.NET
- Typescript: Cockatiel eða Polly-js
- Java: Resilience4j

Í samskiptum við API þjónustur skal nota eftirfarandi forgang þegar samskiptalag er hannað

- Nýting á OpenAPI skilgreiningum við sjálfvirka smíði á forritunarkóða (e. code generation). Þetta á sérstaklega við framendakóða.
- Nýting á forritunarsöfnum sem hjúpa HTTP samskipti, t.d. fyrir C# þá ber að skoða að nota Flurl, Refit eða Resharp.
- Í allra síðustu lög skal útfæra samskipti beint á móti HTTP klösum í Java og .NET (sbr HttpClient í .NET).

Viðmót og vefir

Leitast skal við að hönnun og forritun á vef og app framendum fylgi hönnunarmynstrum sem draga lærdóm af micro-frontend högun og aðgreini skýrt mismunandi aðgerðarsvið innan framendalausnarinnar.

Þjónustur

Við tökum ekki við hugbúnaði til rekstrar sem samsettur er úr smáþjónustum (e. micro/nano services). Bakendakóði skal vera útfærður samkvæmt þjónustuhögun (e. service orientated architecture) en með áherslu á stærri þjónustueiningar.

Lagt er til að högun bakendaþjónusta miðist við lóðrétta sneiðar (e. vertical slice architecture) frekar en láréttar lagskiptingar þar sem hentar. Hvatt er til að skoðuð sé REPR[2] (e. request endpoint response) högun fyrir hefðbundna WebApi högun frekar en MVC haganir (e. model-view/viewmodel-controller).

Huga skal að því að nota viðeigandi tímabundnar gagnageymslur (e. caching) í þjónustum á skipulagðan hátt til að flýta fyrir afgreiðslu og draga úr álagi á gagnagrunnskerfum þar sem mögulegt er.

Allar þjónustur skulu hafa OpenAPI skilgreiningar (forritunarskilgreiningar) sem innihalda skjölun á endapunktum, breytum og gildum sem tekið er við og skilað er. Einnig skal OpenAPI skilgreiningin innihalda dæmi um innsend og skilagögn (e. request and response). Skilgreiningin skal líka skjala villukóða og skilagildi þeirra. OpenAPI skilgreiningin skal vera nógu nákvæm til að hægt sé að nýta sjálfvirk kóðagerðatól (e. code generation).

Gögn sem eru send og móttekin af þjónustum skulu vera á JSON sniði ef mögulegt er. Innri þjónustur geta nýtt sér gRPC/Protobuf í samskiptum sín á milli ef við á. Ekki skal gera greinamun á milli stórs og lítils stafs í svæðaheitum í JSON skeytum (t.a.m. „data“, „Data“, „DATA“, „dATa“ eru allt sama svæðið).

Skýrt skal vera í OpenAPI skjölun hvaða auðkenningarmáti er notaður í forritunarskilum þjónustunnar, hvaða aðgangspunktar eru aðgangsstýrðir og hverjir ekki. Einnig skal vera hægt að prófa þjónustuna á sjálfvirkan máta óháð ytri auðkenningarþjónustum.

Skýrt skal vera í OpenAPI skjölun hvaða aðgangsheimilda er þörf fyrir hvern endapunkt í forritunarskilum þjónustunar.

Aðgreina skal með viðskeyti á nafni á milli þjónusta sem eiga að vera aðgengilegar á internetinu og þeirra sem sitja bak við eldvegg. Þessar nafnareglur eru hugsaðar sem einföld leið fyrir netrekstur og kerfisstjóra til að stilla rétt aðganga að þeim þjónustum sem keyrandi eru. Nota skal -api viðskeyti fyrir þjónustur sem eru aðgengilegar internetinu, viðskeytið -service skal nota fyrir þær sem sitja fyrir innan eldvegg. T.d. auditing-service og auditing-api. Verklagsreglan er að netöryggisaðilar geti unnið eftir þeirri einföldu reglu að „eitthvað -service“ má aldrei vera aðgengilegt utan frá. *Sjá nánari umræðu um viðskeyti í kaflanum um Skipulag kóðageymsla.*

[2] https://www.apitemplatepack.com/docs/introduction/repr-pattern/
https://deviq.com/design-patterns/repr-design-pattern

Forritunarkóði

Allur kóði skal fara í gegnum samræmingar og stílaðlögunar forrit (e. linting). Við notumst við SonarCloud og SonarLint til að samræma kóðastíl í öllum forritunarmálum.

- https://www.sonarsource.com/products/sonarlint/ide-login/
- https://www.sonarsource.com/products/sonarcloud/

Til viðbótar þá eru eftirfarandi sérforrit notuð til að tryggja samræmi og öryggi í kóða (ath að hægt er að slökkva á ákveðnum stíl reglum ef þörf er á í verkefninu, slíkt þarf þó að gerast í samráði við sérfræðinga okkar).

- C#: Roslyn Analyzers (StyleCop) með global.ruleset stillingum frá okkur sem grunn.
- Javascript (nodeJs, Typescript etc): ESLint með eslint-config-airbnb og eslint-config-airbnb-typescript reglusettunum sem grunn.

Öll samræmingar og stílaðlögunarforrit skulu vera stillt á þann hátt að viðvaranir séu meðhöndlaðar sem villur (e. treat warnings as errors).

Mælst er til að hugbúnaðarteymi sinni reglulegu gæðaeftirlit, t.a.m. séu með reglulegt kóðarýni, á meðan að á verkinu stendur til þess að unnið sé jafnt og þétt að því að tryggja gæði, öryggi og stöðuleika afurðarinnar.

Forritarar skulu gæta að forrita í sama kóðastíl og viðkomandi lausn hefur nú þegar, eða í þeim stíl sem sambærilegar lausnir hjá okkur bera. Með kóðastíl er ekki verið að vísa bara í hluti sem linterar geta höndlað heldur almenna högun og arkitektúr á lausninni og einingum hennar. Ekki skal gera breytingar á kóðastíl lausnar án fyrra samtals og samþykki frá sérfræðingum okkar.

PR skulu ekki innihalda breytingar sem eru ótengdar þeirri vinnu sem var verið að framkvæma, eða innihalda breytingar í ótengdum hlutum kóðasafnsins.

Pakkar og namespaces skulu byrja á “Directorate.Health.PROJECTNAME” eða “Directorate/Health/PROJECTNAME” eftir því sem við á. Þar sem *PROJECTNAME* er nafnið á verkefninu (eða stytt útgáfa þess).

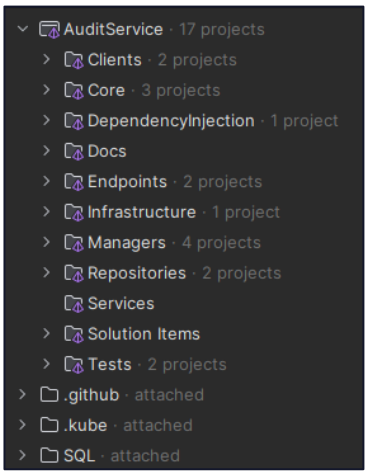
Virknisflögg (e. Feature-flags)

Heimilt er að nota virknisflögg í kóða. Slík notkun er þó háð þeim skilyrðum að annað hvort eru notuð innbyggð virkni í forritunarsafnið til að útfæra virknisflögg (sbr. FeatureFilter í .NET) eða með

forritunarstýringum (e. macros) sem fjarlægja ákveðnar kóðagreinar í build ferlum fyrir PreProd og Production umhverfi.

Lagskipting og skipulag lausna

Hver lausn þarf að fylgja eftirfarandi rótarmöppu (e. top level) uppskiptingu fyrir forritunarkóða. Skiptingunni er ætlað að framfylgja æskilegri arkitektúr högun og lagskiptingu. Enginn forritunarkóði skal liggja beint undir þessum yfirmöppum heldur vera skipulagður í undirmöppur eftir því sem við á.



Dæmi um rótarmöppu uppskiptingu í bakenda .NET verkefni er að sjá hérna til hægri.

Í ákveðnum tilvikum þá getur forritunarkóði verið geymdur í yfirmöppu sem heitir þá „src/“.

Mappa/Skrá	**Á við um**	**Lýsing**
Clients/	Bakendi, framendi (SSR)	Mappa, inniheldur þau project sem útfæra entry pointa inn í hugbúnaðinn (sbr. exe eða entry dll).
Core/	Bakendi, framendi	Mappa, kjarna virkni og innsta lag domain lagskiptingarinnar. Hvatt er til að allir contract klasar (sbr. request/response) séu aðgreindir í sér möppu/project. Hér hvílir einnig hjúpun á statískum framework kóða og sameiginleg configuration og interfaces.
DependencyInjection/	Bakendi, framendi	Mappa, inniheldur kóða sem tengist DI og IoC. Skráningu á einingum og týpum.
Docs/	Bakendi, framendi	Mappa, inniheldur skjölun fyrir lausnina og allar myndir sem henni tengjast.
Endpoints/ eða Controllers/	Bakendi	Inniheldur kóða sem meðhöndlar ysta lag beiðna og fyrirspurna (request / response). Ef REPR högun er notuð þá skal nota orðið „Endpoints“, ef MVC högun er notuð skal nota „Controllers“. Mikilvægt er að skrár hér séu flokkaðar á skynsaman hátt í möppur til að forðast óþarfa flækjustig.
Pages/ eða Modules/	Framendi	Eiginlegar „síður“ eða viðmót í framendalausninni.

Components/	Framendi	Endurnýtanlegir viðmótshlutar innan lausnarinnar.
Hooks/	Framendi	Endurnýtanlegir hooks, t.d. ef notað er React.
Infrastructure/	Bakendi, framendi	Hjúpun á ytri forritunarsöfnum. Inniheldur project sem ættu að byrja á forskeytinu „Library.“, hjúpar þriðja aðila forritunarsöfn sem notuð eru í lausninni.
Managers/	Bakendi, framendi	Flæðistjórnunar kóði í lausninni. Límið milli Endpoints/ og Repositories/. Inniheldur klasa sem lýsa aðgerðarflæði í kerfinu, nota Repository til að sækja gögn og vinna áfram með þau og umbreyta í domain módel.
Repositories/	Bakendi, framendi	Gagnasókn í ytri kerfi, inniheldur project til að sækja í gagnagrunna (Project með endingu „DataAccess“), biðraðir („StreamAccess“, „QueueAccess“ eða sambærilegt), ytri API skil („ApiAccess“), flatar skrár („FileAccess“) eða aðrar gagnageymslur. Vinnur með data módel klasa en ekki domain módelið.
Services/	Bakendi, framendi	Þjónustuklasar, t.a.m. bakgrunnsþjónustur, möppunar þjónustur eða álíka sértæk virkni sem notuð er af Manager lagi en ekki beint af öðrum.
SQL/	Bakendi	SQL kóði sem fylgir lausninni, hér er viðhaldið SQL kóða til að búa til gagnagrunn og alla hluti í honum frá grunni. Einnig eru hérna migration scriptur ef við á. Mikilvægt er að skipta skjölunum upp eftir tilgangi og númera, sbr, 1-create-db.sql, 2-create-tables.sql, 3-create-indexes.sql, 4-users-and-logins.sql, 5-roles-and-grants.sql, 6-assign-roles-to-users.sql osfrv.
Tests/	Bakendi, framendi	Inniheldur einingaprófana möppu ásamt innleiðingarprófunum, viðmótsprófunum og öðrum tegundum af prófunum. Hér má einnig geyma afrit af handvirkum prófunum ef við á.
.github/	Bakendi, framendi	Sjálfvirkni tengd því að keyra prófanir og gæðaeftirlit á kóða, auk sjálfvirkar útgáfu.
.kube/	Bakendi, framendi	Argo CD innviðakóði, settur upp í sömu möppu uppbyggingu og gitops innviða repo.
.gitignore	Bakendi, framendi	Hvaða skrár eiga ekki að fara í git.
.dockerignore Dockerfile	Bakendi, framendi	Docker pökkunar skipanir
SonarLint.xml global.ruleset	Bakendi, framendi	Linting og stillingar á gæðaeftirliti kóða
README.md	Bakendi, framendi	Lýsing á hvað kóðageymslan hefur að geyma.

Þýðingar

Almenn regla er að allar framendalausnir skuli geta birt viðmót sín, myndir og skrár á fleiri en einu tungumáli (íslensku og ensku að lágmarki). Á sú krafa við um allan texta á stýringum, í leiðbeiningum, titlum, villum og öllum öðrum texta sem birtur er notandanum eða er ætlaður honum til yfirlestrar. Þegar skipt er um tungumál skulu einnig allar framsetningar á gögnum fylgja reglum viðkomandi tungumáls eða landssvæðis (sbr dagsetningar, vikudagar, tími, tölur og upphæðir).

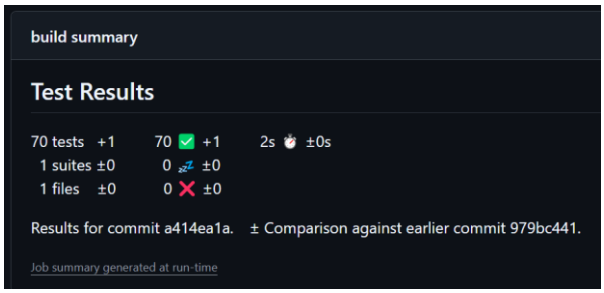
Mælst er til að framendar byggi upp þýðingarskrár sínar þannig að skýrt er hvar þýðingafasti er notaður í viðmótum. Þumalputtareglan er að nota forskeyti sem endurspeiglar nafn viðkomandi síðu eða einingu sem birtir viðkomandi texta. T.d. fyrir síðu sem heitir `viewPatient` þá myndi hnappurinn `close` á þeirri síðu nota þýðingafastann `viewPatient.close`

Almennt er ráðlegt að nota ekki táknmyndir eða myndefni sem inniheldur texta sem þarfnast þýðinga.

Prófanir

Einingaprófanir (Unit tests) skulu fylgja öllum forritunarkóða

- Einingaprófanir skulu aldrei þekja minna en 80% af nýjum kóða þjónustunnar og heildar þekja skal aldrei fara undir 65%. Allar meginlínur þjónustufalla skulu vera prófaðar af a.m.k. einu einingaprófi sem staðfestir virkni aðgerðar og öðru einingaprófi sem staðfestir villumeðhöndlun aðgerðar
- Heimilt er að útiloka ákveðinn kóða frá einingaprófunum í samráði við sérfræðinga embættisins.
- Einingaprófanir skulu vera geymdar sér undir möppu Tests/ skipt upp í sér project (C#), sér rótar pökkum (Java) eða í sér rótar möppum/namespace (önnur mál) eftir því hvers tegundar prófanirnar eru (unit tests, integration tests, ui tests, manual tests)
- Uppbygging einingaprófanna skulu endurspegla möppuskipulag lausnarinnar, en hafa rótarmöppuna Tests/UnitTests/
- Uppbygging einingaprófanna skulu endurspegla ytra skipulag lausnarinnar (url path), en hafa rótarmöppuna Tests/IntegrationTests/

- Niðurstöður einingaprófana skulu vera teknar saman og liggja skýrt fyrir í niðurstöðum úr sjálfvirkum GitHub Actions ferlum. Í það minnsta skal keyrslan innihalda „Test Results“ job sem tekur saman niðurstöður einingaprófana á skýran og hnitmiðaðan hátt. Dæmi um slíkt:
- 

- Prófunargögn fyrir einingarprófanir skulu fylgja forritunarkóða og vera hluti af einingaprófunar projecti. Prófunargögn skulu sitja undir rótarmöppu verkefnisins í Tests/UnitTests/TestData og endurspegla möppuskipulag lausnarinnar
- Öll forritunarskil (APIs) skulu innihalda viðeigandi samþættingaprófanir fyrir alla endapunkta sem eru aðgengilegir
- Það skal vera hægt að prófa forritunarskil þjónusta á sjálfvirkan máta óháð ytri auðkenningarþjónustum
- Einingapróf skulu vera skrifuð í viðurkenndu einingarprófunar frameworki, s.s. xUnit, jUnit eða sambærilegu. Ekki er leyfilegt að nýta heimasmíðað eða óreynt framework til einingaprófana sem hluta af hugbúnaðarlausninni né nota eitthvað sem var smíðað af verktaka og ætlað fyrir önnur verkefni
- Einingapróf skulu fylgja nafnahefðum eins og þær eru skilgreindar af Microsoft hér https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices#naming-your-tests Nöfnin skulu vera samansett af þremur hlutum, sbr „Add_SingleNumber_ReturnsSameNumber“
    - Nafnið á fallinu sem verið er að prófa „Add“
    - Tilvikið sem verið er að prófa, „SingleNumber“
    - Væntanleg niðurstaða úr prófinu, „ReturnsSameNumber“

Skjölun í forritunarkóða (e. comments)

- Comment, forritunarkóði og leiðbeiningar fyrir forritara skulu skrifuð á ensku
- Vanda skal málfar og gæta þess að persónulegar skoðanir eða formælingar eiga ekki heima í forritunarkóða

- Að lágmarki skulu klasar, föll, fastar, enum, osfrv, sem eru merkt “public” vera skjöluð með upplýsingum sem lýsa tilgangi viðkomandi einingar, auk sambærilegrar skjölunar á svæðum skilagildum og inntaksgildum og villuskilyrðum
- Enum og fastar skal hafa skjölun fyrir hvert gildi ásamt lýsingum á virkni þegar gildi er valið.
- Leitast skal við að comment lýsi afhverju kóðinn er gerður á þann hátt sem hann er gerður en ekki hvað hann er að gera.
    - Flott: `//Confirm that the user has read privileges to the data`
    - Slæmt: `//Call privilege service`

Endurnýting kóða / forritunarsöfn

Við hvetjum alla verksala til að nýta sér fyrirliggjandi endurnýjanleg kóðasöfn sem hafa verið smíðuð fyrir embættið og bjóða upp á að auðvelda útfærslu á stöðluðum hlutum sumum sem rætt er um í þessari þróunarhandbók.

Forritunarsöfnin eru gefin út í lokuðu Github Package Registry. Hægt er að fá upplýsingar að aðgangi að þessu safni frá sérfræðingum eða verkefnastjórum embættisins.

Einnig er hægt er að fá aðgang að kóðanum fyrir söfnin á Github embættisins. Hægt er að finna söfnin ef leitað er að „package-“ https://github.com/MRH-Landlaeknir?q=package-&type=all

Ef verksali ákveður að nýta pakka eða forritunarsöfn sem verksali sjálfur á eða hefur smíðað fyrir verkefnið (t.a.m. nuget eða npm pakka) þá þarf útgáfan af viðkomandi forritunarsafnskóðanum sem notaður er í lausninni að fylgja skilum og vera vistaður í kóðageymslur EL. Fyrir útgefna pakka (e. public á nuget.org) þá nægir að fá afrit af viðkomandi útgáfu af nuget pakkanum en fyrir alla óútgefna eða einka pakka (e. private) þá þarf kóði fyrir viðkomandi útgáfu að fylgja með skilum á lausninni. Þetta er til að tryggja stöðugleika í CI pípunum okkar ef ske kynni að pakkaútgáfur verðir af einhverjum ástæðum óaðgengilegar.

Öryggi og auðkenningar

Atriði í þessum kafla ræða einungis ákveðin valin tæknileg atriði tengd öryggi, auðkenningu og heimildaveitingu. Ekki er um að ræða heildarkröfur til öryggis heldur einungis stiklað á grunnkröfum sem tengjast hugbúnaðarsmíðinni beint. Heildarlista yfir öryggis og auðkenningarkröfur er að finna í öryggiskafla í gæðahandbók Landlæknis og kröfur í útboðsgögnum (ef við á).

Almennt um öryggi

- Kerfi skulu geyma allar auðkenningarupplýsingar á öruggan hátt í keyrsluumhverfi.
- Lykilorð og auðkenningarupplýsingar skulu vera send inn í kerfið við ræsingu. T.d. með Hashicorp Vault eða álíka lausnum.
- Nota skal staðlaða dulkóðun á öllum samskiptum. Ekki má útfæra heimasmíðuð dulkóðunaralgrím.
- Dæmi um dulkóðunarstaðla á samskiptum: TLS, HTTPS (með TLS), VPN
- Kóði sem útfærir auðkenningarvirkni skal vera í loka útgáfu (e. final/stable release). Kóði sem útfærir auðkenningu má ekki vera í alpha eða beta útgáfum (e. pre-release). Á þetta sérstaklega við um ytri forritapakka og forritasöfn.

Almennt um auðkenningar

- Kerfi skulu nota margþáttaauðkenning (e. multi-factor authentication, MFA) í öllum auðkenningarleiðum sem innihalda viðmót ætluð almennum notendum.
- Notast skal við Javascript Web Token (JWT) þar sem mögulegt er.
- Notast skal við viðurkennd forritunarsöfn til að meðhöndla JWT gögnin, sjá https://jwt.io/libraries
- Kerfi skulu aldrei endurnota auðkenningatókena úr ytri auðkenningakerfum (þ.m.t. Saga og island.is).
- Þegar notaðir eru auðkenningartókenar frá ytra kerfi (s.s. Sögu eða frá Ísland.is) skal eftir sannreyningu ávallt umbreyta þeim í nýjan tóka sem notaður er í framhaldinu af bak og framenda í viðkomandi kerfi.
- Hugbúnaður keyrandi á útstöðvum notenda skal notast við Active Directory heimildaveitingu (e. AD Groups).
- Kerfi skulu bjóða upp á möguleika á að óvirkja auðkenningartóka í neyðartilvikum.

Almennt um aðgangsheimildir

- Þjónustur skulu nota aðgangsheimildarþjónustu Landlæknis þar sem mögulegt og hagkvæmt er.
- Heimildaveiting skal vera byggð á aðgerðarskiptingu (e. claims) en ekki hlutverkaskiptingu (e. roles).
    - Dæmi, PatientSummary, MedicationDispense, TreamentRelations
    - Ekki: Administrator, Doctor, Staff osfrv.
- Aðgerðarskipting skal vera stöðluð og sniðin að `xxx-Read` og `xxx-Write` aðgerðum.
    - Dæmi, PatientSummary-Read, MedicationDispense-Read, MedicationDispense-Write, TreatmentRelations-Read

- Forðast skal að hafa annað en Read og Write aðgerðir, ef frekara heimildaniðurbrots er þörf skal nota forskeyti á forminu `xxx-yyy-Read` og `xxx-yyy -Write`
    - Dæmi, TreatmentRelations-Add-Write, TreatmentRelations-Remove-Write

Framendar

- Skulu nota rafræn skilríki þegar til auðkenningar á notendum.
- Ef rafræn skilríki eru ekki möguleg skal notast við aðrar viðurkenndar MFA / 2FA lausnir í heilbrigðisþjónustu.
- Vefkerfi sérstaklega
- Nota skal *httponly session cookie* fyrir auðkenningarupplýsingar (s.s. JWT eða session-id)
- Ef notaður er JWT token þá má skipta honum í tvennt og skrifa í tvær kökur, annars vegar í heimildaveitingarhluta (e. payload) og auðkenningarhluta (e. signature). Bakendi fær báðar kökur og setur saman sín megin. Þannig er hægt að hafa aðgang að heimildaveitinga hlutanum í Javascript kóða en auðkenningu örugga.
- Ekki má geyma auðkenningarupplýsingar í local storage í vafrara.
- Takmarka skal aðgang Javascript kóða að auðkenningarupplýsingum.
- Ekki má geyma heilbrigðisupplýsingar eða aðrar viðkvæmar persónuupplýsingar í local storage í vafrara.

Bakendaþjónstur

- Ef ekki eru notaðir JWT þá skal notast við ApiKeys (eða sambærilegt) fyrir samþættingarsamskipti milli kerfa.
- Apikeys skulu bjóða upp á að læsa þeim á ákveðnar IP númera raðir. Ef slíkt er ekki hægt þá skal notast við VPN tengingar.
- Útfæra skal throttling per auðkenningu á ógildum auðkenningarfyrirspurnum og sambærilegum fyrirspurnum.
- Til að hindra DDOS árásir, skal ekki vista ógildar auðkenningarfyrirspurnir í gagnagrunni.
- Fyrir ógildar auðkenningarfyrirspurnir skulu þjónustur forðast að læsa aðgangi heldur lengja í sífellu tímann sem verður að líða milli auðkenningarfyrirspurna, sbr fyrsta ógilda = engin-töf, svo 2 sec, 4 sec, 8 sec, 16 sec osfrv.
- Setja skal takmarkanir á fjölda sambærilegra aðgerða sem leyfðar eru á sekúndu fyrir hverja aðgerð. T.d. ætti ekki að vera leyfilegt að biðja sama endapunkt um að framkvæma sömu aðgerðina 100 á sömu sekúndu/mínútu.
- Notast skal við stöðluð forritunarsöfn eða innbyggða virkni í .NET við útfærslu á takmörkunum í öryggisskyni.

Gagnagrunnar

- Auðkenningarupplýsingar, s.s. password-hashes eða API keys skulu geymdar í dulkóðuðu og aðgangsstýrðu töfluskema sem aðgreint er frá öðrum hugbúnaðargögnum.
- Allir gagnagrunnar skulu notast við innbyggða dulkóðun á disk.

Hugbúnaðarprófanir

> 🌞 **Leiðarljós**
>
> Gæði og hagkvæmni prófana á hugbúnaði er jafn mikilvægar okkur og lausnarkóðinn sjálfur. Ómögulegt er að byggja upp öryggi í framtíðauppfærslum og breytingum án fjárfestingu í góðum gæða prófunum frá byrjun verkefnis. Það er stefna okkar að hugbúnaðarlausnir okkar verði í rekstri og viðhaldi í fjölda ára. Með þetta að leiðarljósi er mikilvægt að allar prófanir sem fylgja hugbúnaðarlausnum okkar séu bæði skrifaðar og skjalaðar á skýran hátt.

Almennar uppsetningakröfur

Prófunarumhverfi

- Við veitum eftir þörfum aðgang að þeim gögnum sem þarf til þess að verktaki geti komið upp prófunarumhverfi sínu

Staðsetning prófana

- GDPR reglugerðin krefst þess að öll gögn landlæknis þurfi að vinna með innan EEA og því er sú krafa gerð að prófanir fari fram innan EEA

Umsýslukerfi fyrir prófanir (test management system)

- Verktaki þarf að vera með umsýslukerfi fyrir prófanir
    - Kerfið þarf að vera aðgangsstýrt
    - Innskráningar, breytingar og keyrslur þurfa að vera skráðar

Utan umhald með villum í þróun

- Sýna þarf fram á að villur séu skráðar, leystar og prófaðar á ný

- Þetta kerfi má vera aðskilið eiginlega umsýslukerfi prófana

Samhengi krafa og prófanna

- Nauðsynlegt er að það komi skýrt fram í prófunargögnum hvaða krafa fylgir hverju prófi
- Hver krafa þarf að hafa að lágmarki 1 skráð próf

Prófanir

Sjálfvirkar prófanir

Öll sjálfvirk próf skal geyma í þar til gerðri möppu sem hlýtur sömu nafnareglum og er talað um í kaflanum um forritunarkóða. Við setjum okkur ekki á móti því að sjálfvirk próf séu keyrð en það er á ábyrgð verktaka að skrifa þær á þann hátt að þær fylgi fyrir framtíðar uppfærslur.

Einingaprófanir (e. unit tests)

- Lágmarks þekja (e. coverage) prófanna fyrir nýjan kóða er 80% með viðmiðið að allur kóði í lausninni uppfylli >65% þekju viðmið.
  - Þekju prósentan má ekki lækka milli útgáfa.
- Ef kröfum um einingarprófanir er ekki uppfyllt er PR hafnað sjálfvirkt
- Einingarprófanir skulu fylgja eftirfarandi leiðbeiningum frá Microsoft,
  - https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices#naming-your-tests
- Hvert einingapróf skal innihalda amk 3 skýrt aðgreinda hluta, aðgreina skal hlutina með eftirfarandi kommentum.
  - `// Setup`
    - Uppsetning á mock breytum og hegðunum, hægt að skipta aukalega upp í `// Expecations` og `// Behaviors` ef þörf krefur
  - `// Test`
    - Prófun á fallinu eða virkninni fer fram hér, söfnun á skilabreytum. Notast skal við nöfnin `underTest` fyrir klasann sem verið er að prófa og `observed` fyrir skilagildi úr falli sem prófað er.

- // Verify
    - Sannreyning á skilagildi og mock klösum.

Dæmi um aðgreiningu á hlutum prófunarfalls

```
[Fact]
public async Task TryHandleAsync_Default_WritesOutProblemDetails()
{
    // Expectations
    var expectedException = new InvalidOperationException();

    // Setup
    var mockLogger = new Mock<ILogger<GeneralExceptionHandler>>();
    var testHttpContext = new DefaultHttpContext();


    // Behaviours
    mockLogger.Setup(x => x.IsEnabled(It.IsAny<LogLevel>())).Returns(false);


    // Test
    var underTest = new GeneralExceptionHandler(mockLogger.Object);
    var observed = await underTest.TryHandleAsync(testHttpContext, expectedException);

    // Verify
    observed.Should().BeTrue();
    mockLogger.Verify(x => x.IsEnabled(It.IsAny<LogLevel>()), Times.Once);
}
```

Samþættingarprófanir (integration tests)

- Öll föll í ytri þjónustuskilum skulu vera prófuð af a.m.k. einu samþættingaprófi sem staðfestir virkni aðgerðar (e. success criteria) og öðru samþættingaprófi sem staðfestir villumeðhöndlun aðgerðar (e. failure criteria)
- Halda skal samþættingarprófunum í lágmarki og einblína á að prófa virkni og réttleika samþættingaskila í kerfinu (sbr. samskipti við aðra APIs, gagnagrunna eða kerfi).
- Samþættingapróf skulu ekki krefjast handvirkar uppsetningar á gagnagrunnum eða gagnasettum til að virka. Nauðsynlegt er að hægt sé að keyra þau sem hluta af sjálfvirkum ferlum og að keyra mörg samhliða samþættingapróf.

Sjálfvirkar notendaviðmótsprófanir (UI)

- Leitast skal við að halda viðmótsprófunum í lágmarki til að halda niðri langtíma viðhaldskostnaði.

- Viðmótsprófanir skulu alltaf keyra á mock bakendum og gögnum.
- Halda skal flækjustigi viðmótsprófana í algjöru lágmarki og einblína á að prófa einungis hluta sem eru nauðsynlegir fyrir virkni kerfisins.
- Nota skal Cypress eða Playwright við smíði viðmótsprófana nema eðli verkefnisins krefjist annars.

Aðgengisprófanir (e. accessiblity testing)

- Notast skal við sjálfvirkar prófanir til að tryggja að viðmótshluti lausna uppfylli lágmarkskröfur um aðgengi blindra, sjónskertra og annara notenda sem nýta stöðluð hjálpartæki við tölvunotkun.

Handvirkar prófanir

- Notendaviðmóts próf (UI test)
    - Notendaviðmóts prófanir þurfa að uppfylla þær notendaviðmóts kröfur sem hafa verið settar fram af notendaviðmóts hönnuði við upphaf verkefnisins
        - Í því tilfelli þar sem þessar prófanir eru gerðar handvirkar á slembikenndan hátt (exploratory testing) þarf að gera prófunarskýrslu þess efnis
    - Notendaviðmót skulu einnig vera prófuð með aðstoð skjálesara til að staðfesta að viðmót séu aðgengileg blindum, sjónskertum og öðrum notendum sem reiða á slík hjálpartæki.
- Viðhald handvirkra prófana
    - Verktaki þarf að standa skil á uppfærðum handvirkum prófum við uppfærslur á hugbúnaði.
    - Ekki má eyða prófum sem hafa verið skráð sem kröfupróf
        - Verktaki getur merkt þau óvirk í samráði við sérfræðinga okkar.

Samþykktarprófanir

Samþykktarprófanir ættu að fara fram fyrir afhendingu á vörunni. Þær eru almennt framkvæmdar á starfstöð þess sem prófar, þar sem prófarar eru sérfræðingar eða aðrir ytri aðilar.

- Starfsmaður okkar er viðstaddur þessa prófana keyrslu
- Verkefnastjóri verktaka er viðstaddur þessar prófanir
- Forritari og/eða tæknimaður verktaka er á staðnum ef þess er óskað

Samþykktarprófanir ættu að vera bókaðar með minnst 2 vikna fyrirvara.

Gögn sem þarf að afhenta með kóðanum

- Prófunargögn
    - Við afhendingu er gert krafa um að við fáum afhent öll prófunargögn
        - Prófunarsvítu (test suite) verkefnis
            - Skráðar keyrslur með tímasetningu
        - Möppur með sjálfvirkum prófunum sem fylgja kóðanum
        - Allar skýringarmyndir og leiðbeiningar fyrir prófara.
- Prófunarskýrslur
    - Prófunarskýrsla þarf að berast í beinu framhaldi af samþykktarprófunum eða í hið minnsta 2 sólahringum fyrir útgáfu ef ákveðið hefur verið að sleppa samþykktarprófunum
        - Mikilvægt er að allar skjámyndir og staðfesting á keyrslu séu inní prófunarskýrslunni
        - Prófunarskýrsla er undirrituð af prófara og verkefnastjóra verkefnis eða gæðastjóra prófunar

Aðgengismál

> 🌼 **Leiðarljós**
>
> Eitt af markmiðum okkar er að tryggja gott aðgengi fyrir alla að þjónustu og upplýsingum í þeim kerfum sem við viðhöldum og eigum. Það skiptir sköpum hugbúnaður sé ekki einungis smíðaður með einn þjóðfélagshóp í huga, heldur að hugsað sé til að fatlaðir og fólk sem lifir með algengar skerðingar (t.a.m. litblindu) geti nýtt sér þjónstu þess hugbúnaðs sem gefin er út af okkur á viðunandi máta.

WCAG staðallinn

Fylgja skal nýjustu útgáfu WCAG staðalsins á hverjum tíma, í Október 2024 er útgáfa 2.2 sú nýjasta https://www.w3.org/TR/WCAG22.

Leitast skal við að uppfylla öll skilyrði A og AA stigsins í WCAG staðlinum auk eftirfarandi eftirfarandi sérvalinna AAA stiga:

- 1.3.6
- 1.4.7
- 1.4.8
- 1.4.9
- 2.1.3
- 2.2.3
- 2.2.5
- 2.2.6
- 2.3.2
- 2.3.3
- 2.4.8
- 2.4.10
- 2.4.12
- 3.3.6
- 3.3.9

WAI-ARIA staðallinn

Öll viðmót hönnuð og útfærð fyrir okkur verða að innihalda viðeigandi ARIA merkingar og fullnægjandi upplýsingar til að notendur geti nýtt sér staðlaða hjálpartækni til að nota hugbúnaðinn.

https://www.w3.org/WAI/standards-guidelines/aria/

Önnur atriði tengd aðgengi

Prófa skal lausnir fyrir hverja útgáfu með skjálesara og íslenskum raddgervli. Notast skal við ráðleggingar frá samtökum blindra og sjónskertra á Íslandi þegar miðað er við hvaða skjálesaratækni skuli styðja að lágmarki.

Þegar mögulegt er, skal notast við sjálfvirka ferla til að hafa eftirlit með aðgengisstöðlum í hugbúnaði.

Hugbúnaðarskjölun

🥸 **Leiðarljós**

Hugbúnaður er aldrei betri en skjölunin sem fylgir honum. Markmiðið með skjölun er að búa til það magn af upplýsingum sem fylgir hugbúnaðarútgáfum þannig að hann tryggir sem best að hægt sé að viðhalda

langtíma gæðum og högun lausnanna. Markmiðið er ekki mikill fjöldi blaðsíða eða umritanir á almennri vitneskju, heldur skjalfesting á högun viðkomandi lausnar og þeim ákvörðunum sem teknar voru.

Hvaða skjölun er skilað við afhendingu

Skjölun skal skilað inn í PDF eða Word skjölum og skal samræma útgáfunúmer og hugbúnaðarlausnin sem er skilað. Skjölun sem ætlað er forriturum eða tæknihópum má skila inn sem Markdown skjölum. Reyna skal eftir bestu getu að geyma all upprunaskjölun og upprunamyndir með hugbúnaðarlausninni, t.a.m. í /docs möppu í Github.

Athugið að allar tæknilegar myndir sem notaðar eru við skjölunina þarf að skila einnig sem .png, eða .jpg skrám (best er að geyma þær undir */docs/images* möppu í Github).

- Architecture og system design sem varpa skýrri yfirlitsmynd af kerfinu
    - Myndir sem lýsa helstu ytri tengingum við kerfið og samskiptum (auk samskiptastöðlum, portum osfrv)
    - Myndir sem lýsa tæknistack lausnarinnar
    - Myndir sem lýsa virkni og samtenginga helstu innviða
    - Myndir sem lýsa gagnaflæði í gegnum hugbúnaðinn
- Forsendur og helstu hönnunarákvarðanir (e. architecture design records. ADR). Mælt er með einfaldri nálgun, sbr. https://medium.com/olzzio/y-statements-10eb07b5a177. Hægt er að skrá ADR með því að klára eina málsgrein sem inniheldur eftirfarandi lykilorð:
    - “Í tengslum við”: Virkni (saga eða notkunartilvik), hluti kerfis, kerfishögun o.s.frv.
    - “standandi frami fyrir”: lýsing á eiginleika sem æskilegur var eða upplifun sem óskað var eftir (non-functional)
    - “ákváðum við”: ákvörðunin sem tekin var
    - “hafandi skoðað”: aðrir valkostir sem skoðaðir voru en hentuðu ekki
    - “til að ná fram”: hagurinn af valinu, hvernig valið uppfyllti skilyrðin sem lögð voru til grundvallar
    - “vitandi það að”: lýsing á veikleikum eða takmörkunum sem eru þekktar tengdar ákvörðununni, langtíma/skammtíma, áhrif á aðra virkni eða kerfi

- Sequence rit fyrir þá kjarnavirkni sem útfærð er ásamt lýsingum og athugasemdum. Sérstaklega þarf að gera sequence rit fyrir öll samskipti sem kerfið hefur við ytri þjónustur, og önnur flæði sem forritarar telja að slík rit hjálpi öðrum hugbúnaðaraðilum til þess að skilja betur flæði aðgerða
- Rekstrarhandbók/leiðbeiningar, miðuð við daglegan rekstur lausnarinnar, t.a.m. uppsetning vélbúnaðar (ef við á), innviðaþarfir, keyrsluumhverfi, kubernetes configuration, logga, metrics, monitoring, regluleg maintenance ferli sem þarf að keyra á lausn og gagnagrunni osfrv.

Hvaða skjölun verður að vera í GitHub

Eftirfarandi skjölun lifir með forritunarkóða í kóðageymslum MRH. Athugið að vænst er til að öll uppruna skjölun sé geymd í /docs möppu í Github repoi verkefnisins. Athugið að niðurbrotinu milli skráa lýst hér er einungis leiðbeinandi og frjálst er að sameina eða skipta skjölum eftir þörfum.

- Eitt **README.md** í rót reposins sem lýsir hvað er að finna í því og hvernig lausnirnar hanga saman
- **README.md** skjal fyrir hverja sjálfstæða lausn (þ.e.a.s. lausn sem ætlað er að keyra í production eða er gefin út sem library eða pakki sem aðrar lausnir nota)
    - Stutt lýsing á því um hvaða hugbúnað er að ræða, tilgang hans og hvaða hlutverki í stærri verkefnum hann tilheyrir
    - Tæknileg lýsing og högun, hvaða forritunarmál, útgáfur af frameworkum hann er byggður á (t.d. .NET 9 fyrir .net verkefni, spring boot v5.0 fyrir Java, þannig háttar)
    - Yfirlitsmynd sem sýnir hvernig helstu hlutar hugbúnaðarins vinna saman og tengjast
    - Dependency lýsing, hvaða ytri tengingar, gagnagrunn eða ytri hugbúnað stólar þessi hugbúnaður á. Upplýsingar um hvaða secrets er þörf og VPN tenginga ef við á
- **deployment.md** skrá í rót lausnar eða hverrar þjónustu eða library projects/solutionar (eftir því sem best við á)
    - Deployment description, hvernig lausn er gefin út, lýsing á docker fyrirkomulagi, buildscriptum, ásamt skrefum sem þarf að taka til að búa til production útgáfu af hugbúnaðinum
    - Deployment leiðbeiningar og þarfir til keyrsluumhverfa, skjölun á kröfum sem docker scriptur hafa (port, env breytur, etc), gagnagrunnskröfum, eldveggja eða infrastructure þörfum etc

- **developers.md** skrá í rót lausnar, í hverri þjónustu eða library projects/solutionar (eftir því sem best við á)
    - Leiðbeiningar fyrir nýja forritara hvernig á að koma lausninni upp á forritunarvél og keyra upp til að debugga, þannig að forritari sem kemur lausninni getur keyrt lausnina og debuggað á vélinni sinni
    - Development leiðbeiningar, compatibility matrixur fyrir external library, user secrets, env breytur, aðgangur að kerfum, appsettings etc, property files, env files etc, upplýsingar um hvernig setja skuli upp tengdar þjónustur ef slíkt á við), hvernig eigi að nálgast logga sem þjónusturnar skrifa
- **database.md** skrá í rót hverjar service eða í repo rótinni ef einn gagnagrunnur fyrir alla lausnina
    - Gagnagrunns leiðbeiningar, uppsetning á gagnagrunni, vísun í hvar sql scriptur eru að finna oþh.
    - Helstu gagnagrunnsnotendur sem lausnin þarfnast og schema aðgang sem viðkomandi notandi þarf
    - Öll scheduled database jobs sem eru hluti af lausninni og keyrslu schedule þeirra.
    - Leiðbeiningar um uppsetningar og viðhald á gagnagrunnum. Hvernig á að hreinsa gömul gögn, periodic cleanup etc.

Viðkvæm gögn

Öllum viðkvæmum gögnum sem stofnuð eru í framvindu verkefnisins skal miðlað á öruggan hátt til ábyrgðaraðila verkefnisins sem hluta af skilum lausnarinnar.

Sem dæmi um viðkvæm gögn

- Dulritunarlyklar
- Öryggisskilríki
- Upplýsingar um aðgangstillingar fyrir tengdar þjónustur sem stofnaðar hafa verið
    - (s.s. username, passwords, client_id, api_keys etc)
- IP whitelisting
- VPN uppsetningar

Geymsla og miðlun gagna

Gagnagrunnar og gagnaskrár

🌞 **Leiðarljós**

Gagnaþörf okkar er mikil og eru öll gögn sem við vinnum með í eðli sínu mjög viðkvæm gögn, bæði persónulega og þjóðfélagslega. Áhersla skal vera á að tryggja bæði gæði gagna og kóða, sem og öryggi gagnanna sjálfra, til að auðvelda langtíma viðhald og úrvinnslu þeirra gagna sem verða til í heilbrigðiskerfinu. Einfaldleiki er oftast besta leiðin til að tryggja langtíma gæði.

Meðhöndlun heilbrigðisgagna

- Ef deila þarf heilbrigðisgögnum (líka prófunargögnum) þá þarf að gera það í gegnum öruggar þjónustur (t.d. Signet).
- Ekki deila eða birta gögn í videó fundum, t.d. teams samtölums.
- Ekki geyma heilbrigðisgögn á útstöðvum forritara eða á fartölvum sem ekki eru með dulkóðaðann harðadisk.
- Ekki geyma heilbrigðisgögn í kóðageymslum.

Gagnagrunnar

Hlutverk gagnagrunna skal vera að geyma og miðla gögnum. Gagnaúrvinnsla er ekki æskileg í gagnagrunnum sem liggja undir kerfislausnum (aðrar reglur gilda um gagnavöruhús). Við tökum ekki við gagnagrunnskerfum þar sem mikilvæg eða kjarnavirkni kerfisins er skrifuð í stefjum, sýnum, töflu virkni eða tímasettum keyrslum (e. procedures, views, triggers, scheduled database jobs).

Við nýsmíði skal velja nýlega útgáfu af Microsoft SQL Server (í samráði við sérfræðinga okkar). Í dag rekur embættið mikið af gagnagrunnum í Oracle en til hagræðinga og samræmingar þá viljum við færa okkur í meira mæli yfir í Microsoft SQL Server. Ef nauðsynlegt er sökum eðli verkefnisins að velja aðra tegund gagnagrunns þarf að fá leyfi fyrir því frá sérfræðingum og gæðastjóra.

Gera skal ráð fyrir að gagnagrunnar sem notaðir eru fyrir PreProd og Production umhverfi séu hýstir í vélasal okkar (e. self-hosted). Einungis grunnar í test umhverfi eru hýstir í Azure skýjaumhverfi embættisins.

Forritunarpakkar (e. ORM)

Hugbúnaðarteymi sem eiga samskipti við gagnagrunna skulu velja létta gagnagrunns forritunarpakka (e. ORM, object-relational mapping). Forðast skal ORM pakka á borð við xHibernate, Entity Framework eða SqlAlchemy.

Forðast skal ORM pakka sem virka á þeim grundvelli að gagnagrunnsnotandinn sé með hærri réttindi en SELECT og EXECUTE á það schema þar sem gögnin eru geymd (sbr. Entity Framework, xHibernate). Af öryggissjónarmiðum á gagnagrunnsnotandi sem bakendaþjónustur nýta sér ekki að hafa CREATE, ALTER eða sambærileg aðgangsréttindi sem geta breytt uppbyggingu gagnaschemans í grunninum.

Gæta skal að velja gagnagrunns uppfærslu (e. migration) stefnu sem myndar ekki óhóflegt magn af viðbótar gagnagrunnshlutum, gera erfitt að vinna með gagnagrunnskemað í gagnagrunninum (t.d. töflunöfn innihaldi GUID gildi og þannig háttar) eða er erfitt að innleiða í sjálfvirk CD ferli (e. continuous deployment).

Gagnagrunnshlutir sem ekki skal nota

Við viljum forðast það að fjárfesta í forritunar- og úrvinnslukóða skrifuðum ofan í gagnagrunnana sjálfa. Þessar kröfur miðast því við að tryggja það að stöðugleiki gagna og heilindi gagnagrunna sé tryggður með því að lágmarka fjárfestingu í sértækri gagnagrunnsvirkni.

Því skal ekki notast við *Stored Procedures*, *Functions* eða *Triggers* til að útfæra virkni kerfisins ofan í gagnagrunninum sem þjónustar keyrsluhluta (e. online part) kerfisins.

Ekki skal nota sértæka SQL virkni sem ekki er hluti af stöðluðum gagnagrunnsútfærslum.

Ef nauðsyn krefst þess að eitthvað af þessari virkni sé notuð skal ræða það á stöðufundi með tæknilegum sérfræðingum okkar. Ef leyfi fæst skal skýrt skjalað í skjölun kerfisins allar *Stored Procedures*, *Functions* eða *Triggers* sem notaðar eru, hvar þær eru notaðar, og tilgang þeirra.

Scheduled Jobs / *Database Jobs* skal ekki útfæra til að sinna nauðsynlegri kerfisvirkni á keyrsluhluta (e. online part) kerfisins. Ef gagnagrunnskerfið krefst þess að keyrð sé regluleg virkni (t.d. til eyðingar á gögnum) þarf að skjala slíkar keyrslur og tíðni þeirra.

Surrogate/Technical keys

Mælst er til að gagnagrunnskerfi notist við surrogate lykla (Surrogate key) til að einkvæmt einkenna gögn í kerfum og til að halda stöðugleika í lyklum milli gagnagrunnskerfa, umhverfa og uppfærsla.

Dulkóðun gagna

Notast skal við AES dulkóðun eða aðra sambærilega örugga dulkóðun. Velja skal a.m.k. 256bita útgáfur af dulkóðunar algrímum (eða hærri bitagildi ef þörf krefur til að tryggja langtíma gagnaöryggi).

Huga þarf að dulkóðun í að minnsta þremur mismunandi tilvikum

1. **Í flutningi**: Við sendingu gagna á milli kerfa (s.s. framenda og þjónustu, eða þjónustu í þjónustu) þá ætti notkun á SSL, TLS eða HTTPS að vera nægilegt til að tryggja það öryggi.
2. **Í geymslu**: Þegar gögn eru vistuð á disk eða í skrár ætti dulkóðun á gagnagrunnum (database level encryption) eða dulkóðun á skrám að vera notuð.
3. **Í úrvinnslu**: Í ákveðnum tilvikum þarf dulkóðun á sjálfum gögnum til að hægt sé að samkeyra og vinna gögnin án þess að fórna öryggi þeirra í úrvinnslu (sbr. dulkóðun á kennitölu).

Nánar um dulkóðun í geymslu

Dulkóðun skal vera útfærð á gagnagrunns stigi eða stýrikerfis stigi. Almennt skal ekki notast við dulkóðun á einstaka gildum í dálkum og töflum í gagnagrunni nema kröfur verkefnisins hljóði upp á það eða vegna persónuverndarsjónarmiða.

Nánar um dulkóðun í úrvinnslu

Ef dulkóða þarf gögn í einstaka svæðum í töflu, (t.a.m. kennitölur eða sjúkdómsgreiningar) þá skal notast við dulkóðunarlykil embættisins. Þetta er nauðsynlegt ef hægt á að vera að samkeyra gögnin milli kerfa án þess að afkóða þau fyrst.

Gögn sem þarf einungis til samanburðar við önnur, sbr lykilorð, á ekki að dulkóða heldur geyma sem *hash+passwordsalt*. Notast skal við SHA-512. Bæta skal við fyrir fram ákveðnu gildi við gögnin áður en þau eru hökkuð saman . Undantekningar á þessu er ef kerfi þarf að viðhalda notendanöfnum eða lykilorðum fyrir ytri kerfi sem ekki er hægt að geyma á annan hátt en að dulrita í gagnagrunnstöflur og geyma.

Útgáfustjórnun

Geyma skal og útgáfustýra skal öllum gagnagrunnskóða í GitHub repoi MRH.

Gagnagrunnskóðinn skal geymdur sem SQL scriptur í sama repo og hýsir þjónustukóða bakendaþjónusta. Geyma skal scriptur í rótarmöppu sem heitir */SQL.* Brjóta skal SQL scriptur upp eftir virkni og númera eftir aðgerðarröð (t.d. *01-create-db.sql, 02-tables.sql, 03-views.sql, 04-users-and-roles.sql, NN-grants.sql, NN-*

indexes.sql oþh). Skal þessi gagnagrunnskóði ávallt útbúa þá útgáfu af gagnagrunni og gagnaskema sem hugbúnaðarkóðinn sem fylgir honum gerir ráð fyrir.

Halda má utan um breytingastjórnun (e. migration) milli gagnagrunnsútgáfa í sér möppu undir SQL/. Skulu skjölin vera geymd sem SQL scriptur og vera nefndar eftir útgáfunúmeri. Sbr. *SQL/migrations/2025.05.12.sql*.

Ef margar þjónustur tengjast sama grunni eða ef geymsla SQL scripta er ekki möguleg með þjónustum skal geyma gagnagrunnskóðann í sér Github repo, gæta skal að fylgja nafnareglum útlistuðum í *Nafnareglur á kóðagreinum*.

Útgáfuferli og keyrslustýring

Docker og Kubernetes

> 🌞 **Leiðarljós**
>
> Það er mikilvægt fyrir okkur að keyrsluinnviði kerfa séu einföld og sem einsleitust. Okkar stefna er að tryggja öryggi með hefð og einfaldleika. Mikilvægt er að vinnuferlar keyrsluumhverfa séu hannaðir með það í huga og fylgi þeim stöðlum sem eru í notkun.

Athugið að þessi kafli gerir ráð fyrir að þróunaraðilar eigi samtal við DevOPS teymi eða sérfræðingana okkar til að fá leiðbeiningar um hvernig best sé að gefa út lausnina í umhverfunum sem embættið keyrir. Nauðsynlegt er að eiga þetta samtal snemma í þróunarferlinu (helst áður en forritun hefst).

Ábyrgðir og væntingar

Umhverfunum sem lýst er í þessum kafla eru þau sem embættið hýsir sjálft og býður verksölum upp á möguleikann á að gefa út á. Ekki er ætlast til þess að verksali útfæri neitt af þeim umhverfum sem lýst er hér í sínum umhverfum eða þróunarumhverfum. Vegna flækjustigs umhverfa okkar þá eru ephemeral umhverfi ekki möguleg.

Gæðaferlar og eftirlit sem lýst er hér er einnig útfært og hýst af embættinu og keyrist sjálfvirkt við skil. Ferlunum og kröfunum er lýst hér svo verksali geti gert ráðstafanir við daglega þróunarvinnu til að tryggja að samsvarandi gæðaeftirliti sé sinnt tímanlega til að forðast óþarfa umfram vinnu við verkskil.

Tæknifólk okkar leggur til grunnvirkni þeirrar sjálfvirknivæðingar sem keyrast skal í kóðageymslum MRH við skil og vinnur, við skil eða fyrr, í samstarfi við verksala hugbúnaðinum að aðlaga grunnvirknina að þörfum skilaafurðanna.

Verksali ber sjálfur ábyrgð á að útfæra alla þá sjálfvirkni sem þeir telja æskilega til að sinna daglegum störfum á sínum vinnustöðvum, þróunarumhverfum og kóðageymslum. Einnig ber verksali ábyrgð á að setja upp og viðhalda þeim þróunarumhverfum og sýndarþjónustum (e. mocked services) sem nauðsynlegt er til að sinna daglegum þróunartengdum störfum hjá sínu starfsfólki. s

Almennt

- Lausnir þurfa að vera stöðulausar (e. Stateless)
- Til að hægt sé að loadbalanca á milli margra tilvika þá má ekki viðhalda innri stöðu í þjónustum (in memory).
- Lausnir verða að gera mælingar (e. metrics) og logga aðgengilega eins og lýst er í þessari handbók til að sjálfvirkir ferlar í Kubernetes geti safnað þeim saman

Docker

- **Docker skrár**
  - Leitast skal við að hafa docker einingar eins litlar og hægt er fyrir hraðari build og útgáfu ferla.
  - Reyna að nýta Alpine docker grunneiningar þegar mögulegt og skynsamlegt er.
- **Build Process**
  - Skal vera eins einfalt og mögulegt er
  - Samræmt milli umhverfa (Dev, Pre-Production, Production)
  - Á **EKKI** að keyra build (compile process)
  - Á **EKKI** að keyra prófanir
  - Á ekki að keyra upp hugbúnað sem **root** notanda heldur skal stofna non-root notanda í ferlinu og keyra á honum
  - Notast skal við tög í samræmi við útgáfustjórnun okkar
  - Skipulag útgáfunúmera. Athugið að tögum er úthlutað sjálfvirkt í pípulínum
  - Docker skráin skal innihalda athugasemdir og útskýringar á ensku
- **Uppbygging skráa**
  - Þjónustur skulu ekki skrifa logga í skrár heldur prenta skal allar logfærslur beint út í standard console out. Logfærslum er safnað saman sjálfvirkt í Kubernetes umhverfi
  - Stillingarskrár í container skrám mega ekki innihalda viðkvæmar upplýsingar eins og notendanöfn og lykilorð. Einnig mega tengingarupplýsingar s.s. gagnagrunns tengistrengir eða IP tölur ekki vera geymdar í skránum sem vistaðar eru í docker skránna
- **Netkerfi**
  - Skilgreina skal hvaða port þurfa að vera opin og hvaða þjónustur treysti á þau
  - Almennt skal reyna að halda utan um samskipti innan containera og eins fá port opnuð eins og hægt er

  - Fylgja skal Topology teikning af netkerfi ef notast er við docker networking
- **Startup**
  - Í Readme skrá skal tekið fram hvað þarf að gera til að keyra upp kerfið
  - Hvernig á að byggja upp image
  - Dæmi um command line keyrslu
- **Hvað ber að forðast**
  - Docker images eiga ekki að keyra build fyrir forritunarkóða. Docker skrár þurfa að pakka upp þeim kóða sem þegar hefur verið byggður af pípunum og var prófaður. Docker skrár mega ekki endurbyggja kóða eftir að prófanir hafa verið keyrðar
  - Ekki keyra prófanir, prófanir skal keyra fyrir utan docker einingar
  - Forðast ber að lenda í þeirri stöðu að vera að keyra docker innan í docker (e. dind). http://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/
- **Leyndarmál / Secrets**
  - Engin leyndarmál (e. secrets) tengd keyrsluumhverfi lausnarinnar skal geyma í docker. Keyrsluleyndarmál skal geyma í Vault og senda inn í docker einininguna þegar hún er keyrð upp í Kubernetes
  - Leyndarmál í Vault skulu ekki notuð í build pípum eða vistuð beint inn í docker einingar (t.a.m. með appsettings.json eða .env skrám)
  - Lykilorð eiga ekki að sjást í clear text í loggum eða með “env” commandi ef tengst er inn á container

Kubernetes

- Hýsingarumhverfi okkar notast við Argo CD
- Allar hugbúnaðarlausnir okkar þurfa að geta keyrt í kubernetes (undanskilið þessu eru gagnagrunnar)
- Kubernetes innviðakóði þarf að vera geymdur sem hluti af kóða lausnarinnar, geymt í sér möppu (.kube) sem þarf að endurspegla Argo CD möppustrúktúr
- Lausnir skulu innihalda viðeigandi virkni (e. Probes) til þess að hægt sé að nýta sjálfvirknivæðingu í Kubernetes. Sjá kafla um „Keyrslustöður“, (dæmi um þetta eru /system/alive og /system/ready endapunktar)

- Allar þjónustur skulu prenta strúktúraðar logfærslur beint út í console out, ekki skal skrifa logga í skrár á disk
- Öll keyrsluleyndarmál (s.s. gagnagrunnstengingar, aðgangslyklar, lykilorð og notendanöfn) skulu geymd í Vault. Leyndarmálum skal skotið inn í docker containera rétt áður en þeir eru keyrðir upp

Geymsla leyndarmála í keyrsluumhverfi

Lykilorð, auðkenningarlyklar og notendanöfn eru geymd í Vault kerfinu í Kubernetes umhverfinu okkar. Sjá nánar viðauka um geymslu leyndarmála.

Rekstur og eftirlit

Mælikvarðar og sjálfvirknivæðing

> 🌞 **Leiðarljós**
>
> Mikilvægt er að eftirlit með stöðu og afköstum hugbúnaðarlausna okkar sé gagnsætt og staðlað. Slíkt stuðlar bæði að rekstrarhagræðingum og sjálfvirknisvæðingu. Af þessum sökum er mikilvægt að lausnir safni saman og skili af sér á staðlaðan hátt mæligögnum, keyrslustöðu og villuskrám sem geri bilanaleit og rekstrarvöktun bæði skilvirka og einsleita.

Þjónustustigssamningar (e. SLAs)

Skýr tveggja eða þriggja aðila þjónustusamningur þarf að vera til staðar milli okkar og þeirra sem sinna viðhaldi annarsvegar og hinsvegar rekstrarlegri þjónustu á hugbúnaðarlausnunum.

Í þeim samningi eru skilgreindar uppitíma kröfur kerfisins, viðbragðstími við frávikum og sektir við frávikum sem fara út fyrir skilgreindar uppitímakröfur. Einnig skilgreinir það skjal vinnuferla og samskiptaferla sem skal fylgja þegar frávik koma upp.

Skjalið skal einnig innihalda útgáfudagatal sem tekur skýrt fram þau tímabil sem leyfilegt er að gefa út uppfærslur og sinna viðhaldi á kerfinu sem krefst þess að kerfið liggi niðri til fulls eða að hluta. Einnig skal skjalið innihalda skilgreiningar á tímabilum þar sem útgáfur eru ekki leyfðar sökum stórhátíðisdaga, fría, þekktra álagspunkta eða fyrir fram séðri manneklu í þjónustuteymum (e. brownout and blackout periods).

Skjalið skal einnig innihalda tilkynningaskyldur á þjónustu aðila tengdar tilkynningaskyldu og fyrirvara á tilkynningum tengdar fyrirhuguðum breytingum á stillingum, kerfum, vélbúnaði eða öðru sem tengist rekstri og aðgengi kerfisins.

Mikilvægt er því að þjónustur og kerfi haldi utan um mælingar á eigin afköstum og uppitíma og geri þær mælingar aðgengilegar á stöðluðu sniði fyrir ytri kerfi til að sækja.

Mælingar

Öll kerfi skulu halda lifandi skrá um núverandi keyrsluafköst þess. Staðallinn sem notast skal við er Prometheus (prometheus.io). Hugbúnaðarteymi geta einnig valið að nota OpenTelemetry staðalinn en það er vert að geta að þau gögn munu að öllum líkindum enda í Prometheus gagnageymslum, það takmarkar notkun á nýrri hlutum sem prometheus styður ekki eins og stendur.

Þjónustur þurfa að útfæra eftirfarandi endapunkt til að birta mælingarniðurstöður

- /system/metrics
    - Birtir núverandi stöðu á Prometheus mælingum þannig að hægt sé að skrapa þá.

Keyrslustaða

Þjónustur þurfa að útfæra eftirfarandi endapunkta til að tryggja að sjálfvirknivæðing á pod level sé möguleg innan Kubernetes umhverfisins.

- /system/alive
    - Svarar fyrir hvort þjónustan sé í gangi og hefur ekki hrunið eða er í lás (e. Deadlock). Ef þetta tékk virkar ekki þá mun Kubernetes endurræsa þjónustuna sjálfkrafa.
- /system/ready
    - Svarar fyrir hvort þjónustan sé tilbúin til að taka við beiðnum, þ.e.a.s. nær þjónustan sambandi við aðrar þjónustur og hefur allar upplýsingar sem þarf til að byrja að fá ytri traffík til meðhöndlunar. Ef þetta tékk virkar ekki þá mun Kubernetes ekki senda traffík á þjónustuna (en þjónustan er skilin eftir í gangi).

Loka þarf á alla utanaðkomandi umferð (af internetinu) á alla endapunkta undir slóðinni `/system/*`.

Einnig má aðgreina `/system/*` endapunkta á öðru porti, ef slíkt er gert skal nota portið `9102`.

Útgáfuupplýsingar

Þjónustur þurfa að útfæra eftirfarandi endapunkt sem notaður verður í sjálfvirka útgáfuferla og sannreyningarferla. Þessir endapunktar þurfa að vera aðgengilegir af internetinu.

- /version
    - Notaður til að sannreyna hvaða útgáfa er á hvaða umhverfi til að styðja við sjálfvirkni í útgáfu á þjónustum
    - Birtir útgáfu upplýsingar og upplýsingar um þann feril sem notaður var til að gefa út þessa útgáfu á umhverfinu.
      t.d.

```
{
  "commitSha": "c9a911a85e0b29c4f08bdf547e37376d800981c1",
  "workflowRunId": "304328",
  "version": "2024.08.10"
}
```

Svæðin í /version

- **CommitSha**: SHA númer þess git commits sem notað var til að byggja útgáfuna. Einkennir nákvæmlega hvað var í kóðagrunninum og er innifalið í þessari útgáfu.
- **WorkflowRunId**: ID á þeirri Github Action keyrslu sem útbjó viðkomandi útgáfu. Gefur kost á að rekja prófunartilvik sem keyrð voru og að sanna hvaða gæðaeftirliti hafi verið sinnt fyrir útgáfuna.
- **Version**: Útgáfunúmerið á CalVer formi.

Logging (upplýsinga og villuskráning)

Notast skal við structured logging á JSON formi. Logging skal fylgja Elastic Common Schema (https://www.elastic.co/guide/en/ecs/current/ecs-reference.html) í grunninn og bæta við þeim viðbótum sem skilgreind eru af okkur.

Leitast skal við að kerfi ætlað er að keyri í kubernetes og eða azure, skrái logga út í console glugga en skrifi ekki logga út í skrár.

Allar log færslur verða að innihalda lágmarks rekjanleika upplýsingar (traceid og spanid).

Viðbótar svæði

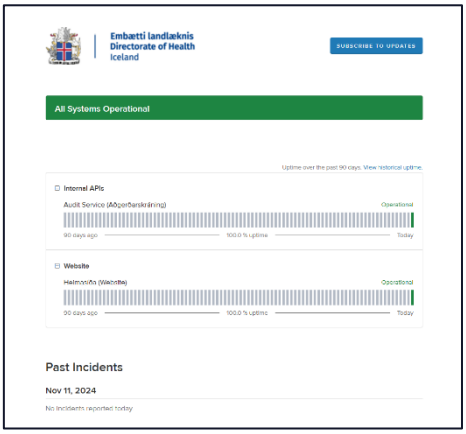
ECS Schema path	Lýsing
labels.x-road-client	Skráning á x-road upplýsingum sem koma inn með x-road-client http header.

labels.x-correlation-id	Skráning á rakningarupplýsingum sem sendar eru inn með x-correlation-id http header.
labels.application	Nafn hugbúnaðarins (þetta þarf að vera einkvæmt nafn). Miðið við að þetta sé lýsandi fyrir kerfisstjóra, ágætis viðmið er að nota GitHub repository nafnið.

Stöðuskráning

Við höldum úti stöðu síðu (e. status page) sem tekur saman yfirlit yfir rekstrarstöðu kerfanna sem við þjónustum. Síðan er aðgengileg á https://landlaeknir.statuspage.io/

Allur hugbúnaður sem smíðaður er fyrir okkur þarf að eiga réttar skráningar á þessari síðu og samþætta uppitíma tilkynningar sínar við hana.



Viðauki 1 – Sjálfvirkir útgáfuferlar í Github

Markmið okkar með sjálfvirknivæðingu í útgáfuferlum er að lágmarka möguleika á mannlegum mistökum, að öll gögn og lykil-/kenni-/leyniorð séu varðveitt og miðlað á öruggan hátt og að ferlar sem nauðsynlegir eru til útgáfu séu skýrt skjalaðir og hægt að endurtaka.

Stefna okkar er að keyra allar sínar lausnir í Docker einingum í Kubernetes umhverfum. Umhverfin verða þrjú, Test, PreProduction (PreProd) og Production (Prod)

- **Test** er hýst í Kubernetes clusteri í Azure skýjaumhverfi okkar. Ætlað fyrir þróun á lausnum og útgáfuferlum, prófanir forritara og aðra tilfallandi vinnu
- **PreProd** hýst í Kubernetes clusteri hjá Sensa. Ætlað til samþættinga- og gæðaprófana á útgáfum.

- **Prod** er hýst í Kubernetes clusteri hjá Sensa. Raunkerfi ætlað fyrir almenning og heilbrigðisstarfsfólk.

Notast er við Argo CD við umsýslu og eftirliti á útgáfum og keyrsluumhverfi lausna. Haldið er utan um allan

Argo CD og Kubernetes stillingar fyrir umhverfi embættisins í .kube/ möppu með lausnarkóðanum.

Fyrirkomulag CI sjálfvirknivæðingar

Haldið er utan um innviðakóða í Git kóðakeymslum skv GitOPS hugmyndafræðinni, er hann geymdur undir .github/ möppu í lausninni.

CI/CD ferlum er stýrt af Github Actions hjá embættinu. Verktökum er frjálst að nýta sér þau CI sjálfvirkni umhverfi sem best henta þeirra vinnustíl í sínum fork (mælst er til að umhverfin séu byggð á Git, sbr Gitlabs, Bitbucket eða Harness).

Github Action Workflowin okkar er að finna undir .github/workflow möppu í hverri lausn. Ef verktakar ákveða að nota einnig Github Actions þá er vert að hafa forskeytið „xternal-„ á þeim skrám til að aðgreina frá okkar skrám (orðið „xternal“ er valið til að skrárnar detti neðst í stafrófsröð). Dæmi

- xternal-pullrequest-opened.yml
- xternal-mergerequest-opened.yml
- xternal-codereview-accepted.yml

Ekki er þörf á forskeytum fyrir aðrar CI þjónustur (sbr .gitlab-ci.yml eða bitbucket-pipelines.yml)

CI pípurnar okkar innihalda stýringar til að hindra að þær séu keyrðar í öðrum umhverfum en okkar. Ef verktakar nýta sér Github Actions þá er mælst til að þær innihaldi samskonar stýringar til að hindra að þau séu keyrð í umhverfi landlæknis.

Dæmi

```
jobs:
  job-not-for-directorate:
    if: github.repository_owner != 'MRH-Landlaeknir'
    steps: ...
```

Við gefum út og viðhöldum sniðmát af Github útgáfuferlum og er það sniðmát aðgengilegt öllum þegar nýjar kóðageymslur eru búnar til.

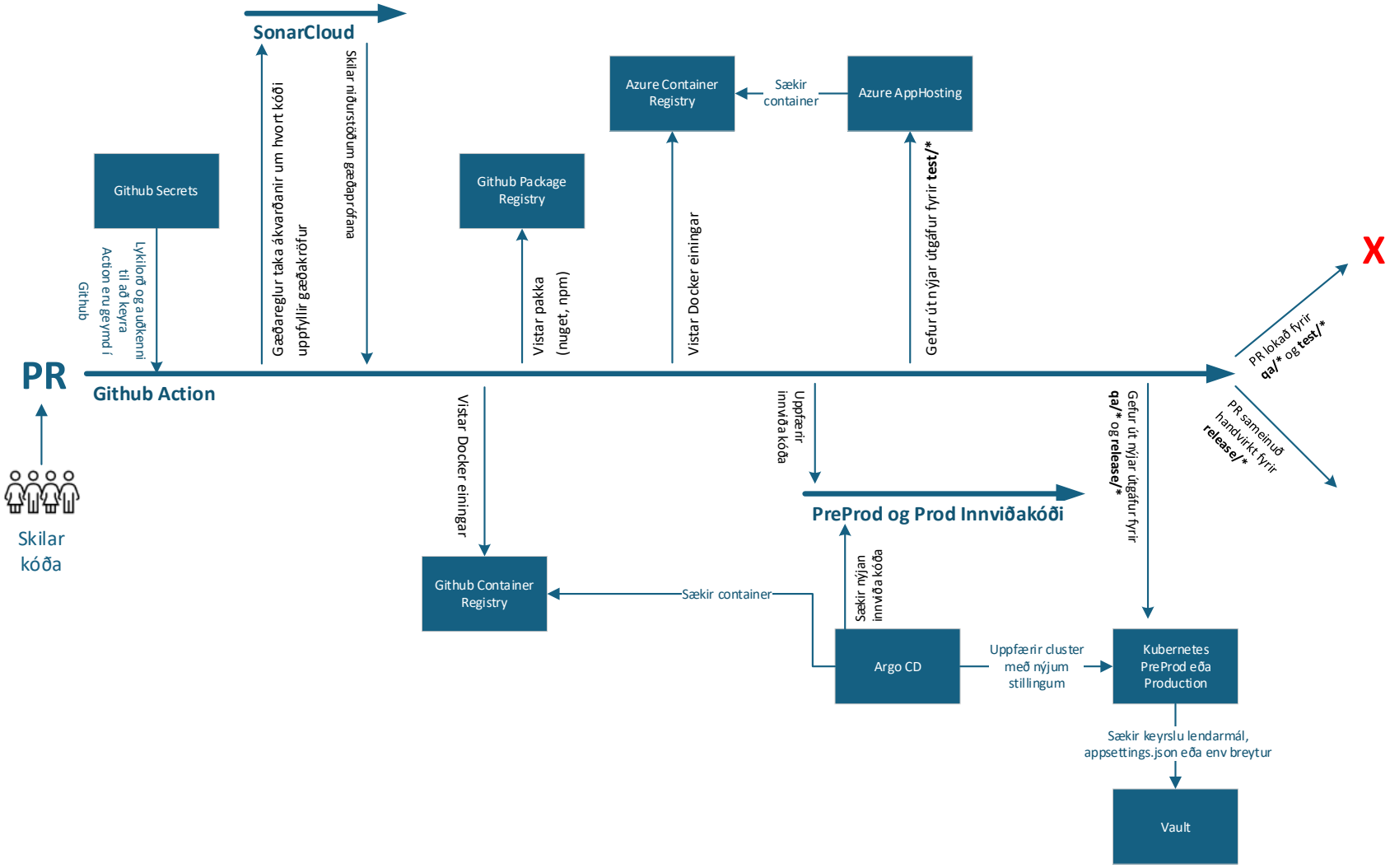
Mikilvæg atriði

- Allar innviðaskilgreiningar og stillingar eiga að vera geymdar sem kóði (e. infrastructure as code). Bæði sem docker skipanir og sem kubernetes skilgreiningar.

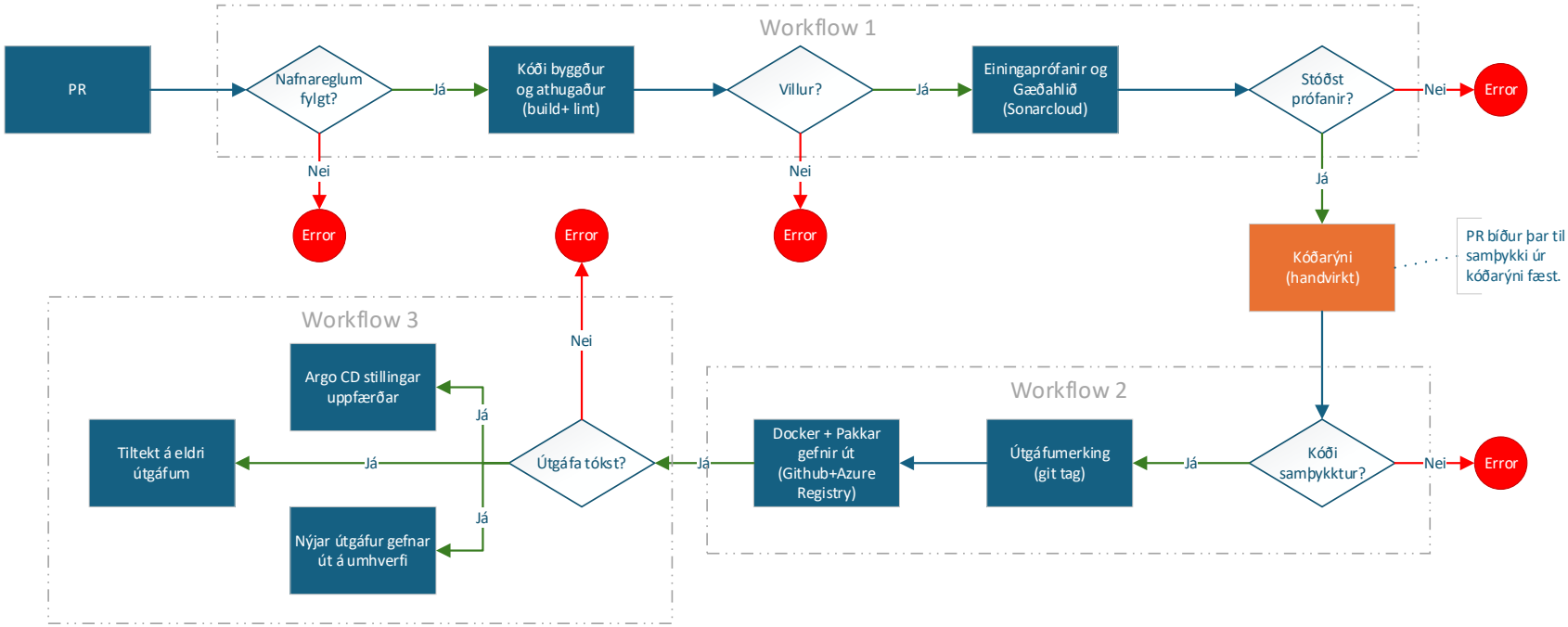
- Github Actions píplínur eiga að sjá um að útbúa kóða fyrir útgáfu (build, compile, test). Leitast skal við að halda Docker skilgreiningum eins einföldum og mögulegt er (helst á docker einingin einungis að pakka inn þeim kóða sem píplínan hefur útbúið og prófað.
- Docker einingar skulu taggaðar bæði með útgáfunúmeri og git commit sha númeri.
- Notast skal við föst docker tög fyrir test keyrsluumhverfi. Þ.e.a.s. latest-dev“ er tagið sem gefið er út fyrir Test.
- Ekki skal notast við föst docker tög fyrir önnur umhverfi.
- PreProd umhverfi skal notast við CalVersion með „-rc“ viðtengingu (sbr 2024.12.09-rc1, 2024.12.09-rc2, 2024.12.09-hotfix1-rc1 osfrv)
- Production umhverfi skal notast við CalVersion með eða án viðtengingu (sbr 2024.12.09 eða 2024.12.09-hotfix1). Ekki skal nota „-rc“ viðtengingu í production.
- Til að tryggja að alltaf séu sóttar nýjar images, þá þurfa K8s poddar að vera stilltir með `imagePullPolicy: Always`
- Reyna að halda píplínukóða viðráðanlegum með því að brjóta upp stór workflow skjöl í smærri afmarkaðri einingar, brjóta upp aðgerðir sem notaðar eru á mörgum stöðum í actions osfv.

Útgáfuferill heildarmynd

Eftirfarandi útgáfuferill



Útgáfuferill í skrefum



Viðauki 2 – Geymsla leyndarmála

Í sjálfvirknipípum í Github skal notast við Github Secrets. Fyrst skal íhuga að geyma secrets á organization stigi (þannig að fleiri en ein kóðageymsla geti samnýtt gildin). Ef slíkt er ekki möguleiki þá skal nota secrets á kóðageymslu stigi.

Í innviðakóða (kubernetes) skal notast við Hashicorp Vault þjónustu sem sett er upp í hverju Kubernetes clusteri fyrir sig.

Fyrir Javascript lausnir, atriði tengd environment breytum

Ef notast er við environment breytur skal kóðinn ekki nota mismunandi nöfn fyrir breyturnar eftir umhverfum. T.d. skal ekki nota API_ URL_PROD og API_URL_PPT heldur einungis API_URL.

Athugið að ef notaðar eru environment breytur þá skal ekki geyma gögn sem ekki teljast leyndarmál í secret geymslum. Venjuleg environment gildi skulu geymast í kubernetes skrám (values.yml) fyrir viðkomandi umhverfi.

Javascript lausnir skulu hlaða inn stillingum úr environment breytum á einum stað í kóðanum og safna saman gildunum í óbreytanlegan (e. immutable) configuration object sem svo er notaður af lausninni. Ekki skal vera að sækja gildi beint úr environment breytum á mörgum mismunandi stöðum í kóða.

Fyrir .NET, atriði tengd Appsettings.json uppsetningu

Öll leyndarmál skulu geymast í appsettings.json. .NET lausnir skulu ekki notast við environment breytur til að sækja stillingar (nema sérstök krafa sé um annað). Leyndarmál skulu geymast sem heil appsettings.json skjöl með öllum stillingum settum.

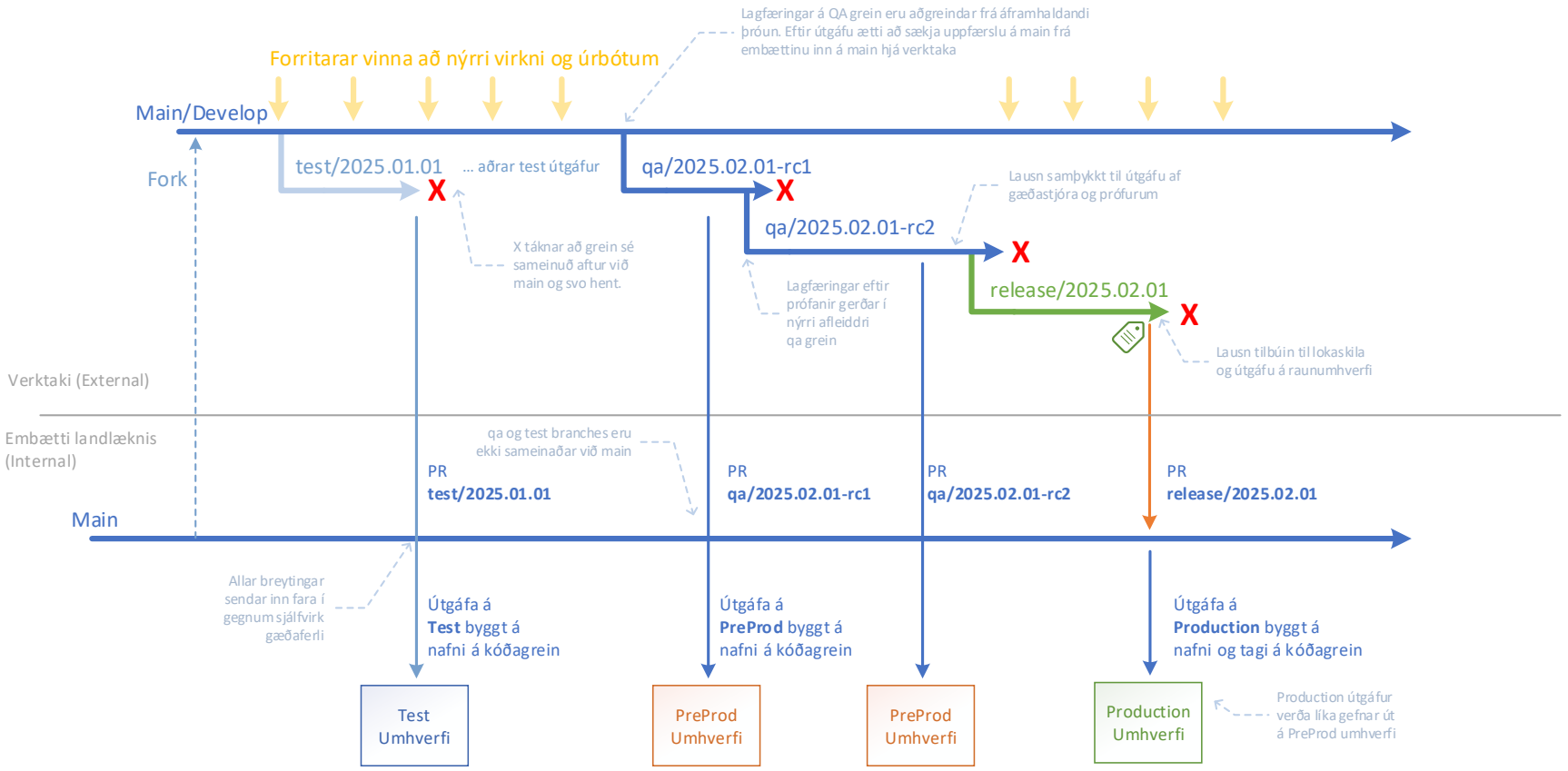
1. Appsettings.json skráin skal geymd í heild sinni inni í Vault umhverfunum (Test, PrProd og Production) með viðeigandi gildum fyrir hvert umhverfi
    a. Lykillinn í vault skal heita `services/data/NAFN-Á-ÞJÓNUSTU/appsettings.json`. Þar sem „Nafn á þjónustu“ er skipt út fyrir það nafn sem valið er.
2. .NET þjónustur skulu innihalda kóða sem reynir að hlaða inn skrá sem heitir `appsettings.override.json` ef hún er til. Sbr program.cs ætti að innihalda kóða líkt og

```
var builder = WebApplication.CreateBuilder(args);
builder.Configuration
       .AddJsonFile("appsettings.json", optional: false, reloadOnChange: false)
       .AddJsonFile("appsettings.override.json", optional: true, reloadOnChange: false)
       .AddEnvironmentVariables();
```

3. Í kubernetes stillingunum skal yfirskrifa **`ENTRYPOINT`** skilgreininguna í docker skránni rétt áður en hún er keyrð upp til að vista appsettings.json gildin úr Vault sem appsetting.override.json inn í containerin. Sbr

```
args: [ "cp /vault/secret/appsettings.json appsettings.override.json && dotnet
Directorate.Health.MY-AWESOME-SERVICE.dll" ]
```

Viðauki 3 – Samspil kóðagreina og skila



Viðauki 4 – Gátlistar við verkefnastýringu

Gátlisti áður en forritun hefst

- ☐ Höfum við heimild fyrir hugbúnaðinum
  - Hugbúnaðurinn og lausnin samræmist gildandi lögum, reglum og fyrirmælum.
  - Lausnin uppfyllir lög og fyrirmæli heilbrigðisþjónustunar á Íslandi.
  - Skjalfest er í README og skjölun verkefnisins hvaða lög, reglur eða fyrirmæli heimila tilvist og þróun hugbúnaðarins og gagnasöfnunar hans.
- ☐ Kerfisþarfir
  - Notendahópar og þarfir þeirra til kerfisins skilgreindir og samþykktir.
  - Skissur af viðmóti liggja fyrir og hafa verið samþykktar.
- ☐ Hönnun og högun
  - Yfirlitsmynd af hönnun kerfisins er tilbúin og samþykkt.
  - Forritunarmál samþykkt.
  - Forritunar framework samþykkt.
- ☐ Varðveisla gagna / Gagnagrunnur
  - Er HL7 staðall sem nota skal skýr, hefur landlæknir skilgreint HL7 sniðið sem nota skal.
  - Ákveðið hvar og hver eigi að geyma gögnin (hvaða gagnagrunnur).
  - Á að sækja gögnin í FHIR þjón landlæknis, er búið að skilgreina hvað á að sækja.
  - Á að vista gögnin í FHIR þjón landlæknis, er búið að skilgreina skema gagnanna.
  - Gagnagrunnsskema (ef við á) er samþykkt.
  - Aðgangur veittur að gagnagrunnskerfi (DBMS eða FHIR)
  - Kóða varðveisla í Github fyrir SQL/gagnagrunnskóða skilgreind og samþykkt.
- ☐ Samskipti milli fram- og bakenda
  - JSON, XML eða ProtoBuf skeyti skilgreind og samþykkt af öllum aðilum.
  - Hvar stillingar eru geymdar og hvernig.
- ☐ Samþættingarþörf
  - Samskipti kerfisins við ytri kerfi skilgreind og samþykkt.
  - Aðgangur að ytri kerfum liggur fyrir.
- ☐ Auðkenning og öryggi

- Auðkenning utanaðkomandi aðila
- Heimildarhlutverk/rullur utanaðkomandi aðila skilgreindar
- Heimildarveiting utanaðkomandi aðila skilgreind
- Auðkenning milli kerfa eða kerfiseininga
- Heimildarveiting milli kerfa eða kerfiseininga
- Aðgangsheimildir fyrir gagnagrunn eru skilgreindar (notendur og roles)

☐ Gæðastýring

- Prófunarkröfur eru samþykktar af öllum aðilum.
- Kóðarýnisreglur samþykktar af aðilum.
- SonarCloud gæðahlið samþykkt af aðilum.
- Undantekningar frá þróunarhandbók liggja fyrir og samþykktar.

☐ Github

- Nöfn og fjöldi Github Repoa ákveðinn
- Uppsetning á GitHub repos er lokið.
- Skilastjóri skilgreindur og hefur aðgang að Github
- Uppsetning á Github Actions fyrir repo lokið.
- Fyrsta „readme“ prófunar PR hefur farið í gegnum skilaferil (PR inniheldur einungis README.md skjal)

☐ Afhendingarform

- Tíðni milli skila skilgreind og samþykkt.
- Fyrirkomulag endanlegra skila er ákveðið og samþykkt.
- Skiladagsetningar sem settar eru af þriðja aðila eru skýrar og samþykktar. (ef við á)

☐ Þróunarhandbók samþykkt og undantekningar skýrar.

☐ Hönnunarhandbók samþykkt og undantekningar skýrar.

Gátlisti fyrir forritun

☐ Uppbygging kóða

- Er högun kóða á disk skýr og samþykkt.
- Kóðahögun fylgir þróunarhandbók
- Undantekningar frá þróunarhandbók tengdar kóða skilgreindar og samþykktar.

☐ Gæðastýring

- Forritarar hafa kynnt sér nýjustu útgáfu af þróunarhandbók.
- Sonarlint (eða sambærilegt) uppsett og virkt í forritunarumhverfum allra forritara.

- Undanþágur frá þróunarhandbók gerðar virkar í config skjölum.

☐ Sjálfvirknivæðing

- CI og CD pípur hafa verið aðlagaðar og samþykktar í .github/.
- Repo uppsett í SonarCloud MRH.
- Argo CD uppsetning hefur verið skilgreind í .kube/ og prófuð.
- Secrets fyrirkomulag skilgreint og útfært.

☐ Docker skrár skilgreindar og útfærðar.

Gátlisti fyrir útgáfustjórnun

☐ Kubernetes þarfir

- Liveness endapunktar skilgreindir og virkir
- Namespace skilgreiningar ef við á

☐ Eftirlit

- Samantekt á loggum og miðlun í Coralogix kerfi.
- Samantekt á mæligögnum (Prometheus) og miðlun til Coralogix.
- Samþætting við status síðu okkar á landlaeknir.statuspage.io

☐ Sértilfelli skjöluð fyrir útgáfuferli fyrir umhverfi

- Test
- PreProd
- Production

Gátlisti fyrir útgáfu á PreProduction og Production

☐ Dagsetningar ákveðnar

- Hvenær lokaskil á kóða fara fram
- Hvenær útgáfa á Preprod/Prod skal fara fram
- [Bara Preprod] Hvenær prófanir hefjast

☐ Skjölun á útgáfuleiðbeiningum

- Lýsing á ytri tengslum og gagnaflæði milli þeirra
- Lýsing á dulkóðunarstandard og hvað er dulkóðað í kerfinu (og hvað ekki)
- Hvernig skal „smoke test“ prófa kerfið (virkar það)

☐ Uppsetning á hýsingarumhverfi lokið

- Uppsetning á innskráningarleið lokið (island.is eða annað)

- Uppsetning á heimildarveitingarkerfi lokið (rullur skilgreindar og notendur sem eiga að hafa aðgang)
- Kubernetes stillingar útfærðar í „.kube“ möppu með kóðanum.
- Tengingar við ytri kerfi virkar og aðgangsupplýsingar á reiðu.
- Tengingar og tengistrengir við gagnagrunna virkar og á reiðu.
- Búið að búa til gagnagrunnsskema á PreProd/Prod gagnagrunnsvélum.
- Búið að taka við kóðanum í GitHub (QA/Release kóðagrein með útgáfunúmeri)

☐ Kóði uppfyllir sjálfvirkar prófanir

☐ Kóði uppfyllir kóðarýni MRH

☐ Endanlegar keyrslueiningar (docker) tilbúnar

☐ Prófanir

- [Bara PreProd] Prófunarhópar skilgreindir
- Hverjir verða að prófa
- Hvað á að prófa, prófunartilvik skilgreind og skjöluð
- Aðgangur fyrir prófara uppsettur
- Skilgreina hvaða tækniaðilar þurfa að vera á vakt á meðan á prófunum stendur

☐ Útgáfudagur

- Skilgreint hvaða aðilar þurfa að vera til taks við útgáfu (tæknifólk, rekstrarfólk)
- [Bara Prod] Uppsetning á rekstrareftirliti
- Skilgreining og uppsetning á sjálfvirkum viðvörunum út frá mæligögnum
- Uppsetning á skröpun á keyrsluloggum fyrir kerfið í miðlægt Coralogix kerfi.

☐ Afhendingarfundur með rekstrarhóp

☐ Rekstrar playbook skilgreind fyrir kerfið og úthringireglur (escalation procedure).

☐ Samþætting við status síðu á landlaeknir.statuspage.io

Gátlisti fyrir verklok

Athugið að þessi gátlisti fyrir verklok vísar einungis í tækniatriði sem vert er að athuga við verklok.

Formlegan gátlista fyrir verklok er að finna í útboðsgögnum og hjá verkefnastjórn.

☐ Endanleg skjölun

- Github inniheldur allar skjölunar skrár sem þörf er á í Markdown skjölum (t.d. README.md)
- Github inniheldur upprunaskjöl fyrir aðra skjölun og myndir (sbr PDF eða WORD) í /docs

- ☐ Rekstrarleiðbeiningar
    - Rekstrarupplýsingar fyrir kerfisstjóra tilbúnar.
    - Sértækar eftirlitskröfur.
    - Sértækar viðvaranir sem þarf að setja upp á eftirlitsgögnum.
- ☐ SLA þjónustustigs- og rekstrarsamningar til staðar og samþykktir.
    - Uppitíma kröfur kerfisins skilgreindar og viðurlög við frávikum.
    - Svartími við frávikum í þjónustu skilgreindur.
    - Frávikaferill og samskiptaferlar skilgreindir.
- ☐ Útgáfudagatal (brownouts and blackouts) skilgreindir

Viðauki 5 – Gátlisti við Kóðarýni

Kóðarýni er gert við skil á hugbúnaðarkóða í QA eða RELEASE kóðagreinum, einungis eru rýnd PR sem eru hluti af qa og release skilum.

Ef kóðarýnir (e. reviewer) metur að eitthvað af áhersluatriðum sé ábótavant er PR hafnað og þess óskað að höfundur geri breytingar til að tryggja að gæðakröfur séu uppfylltar.

Sá sem rýnir skal hafa eftirfarandi í huga við rýnið

- Áhersla á mikilvægi og áhrif breytinganna, ekki stílinn sem notaður er.
- Spyrja spurning og eiga uppbyggjandi samtöl.
- Nýta tækifærið til að kenna, ekki gagnrýna.
- Búa til öruggt umhverfi þar sem óhætt er að gera mistök.
- Sjá tækifæri líka til að hrósa fyrir flottar útfærslur eða nálganir.

Hvað skal skoða sérstaklega í rýni, gátlisti

Athugið: Mikilvægasta er að meta hönnun og gæði þess kóða sem er í rýninu.

- ☐ Er eðlilegt samspil mismunandi kóða í breytingarbeiðninni? Meikar þetta sens?
- ☐ Á þessi breyting heima í kóðagrunninum eða er hún betur geymd sem library?
- ☐ Samþættist breytingin vel við núverandi kerfi eða sambærileg kerfi?
- ☐ Er þetta rétti tíminn til að bæta við þessari virkni?
- ☐ Gerir kóðinn það sem forritarinn ætlast til að hann geri? Er útfærslan rétt?

- ☐ Fylgir kóðinn eftirfarandi hugmyndafræði
  - ☐ SOLID principles + SoC (Separation of concerns)
  - ☐ ONION architecture + Principle of least knowledge / Multilayered architecture
  - ☐ CLEAN code
  - ☐ KISS (keep it simple silly)
  - ☐ DRY (don't repeat yourself)
  - ☐ YAGNI (You aren't going to need it)
- ☐ Er flækjustig kóðans við hæfi? Er kóðinn of flókinn? Er kóðinn of almennur (e. generic)? Línur, föll, klasar of flóknir? Er líklegt að nýjir forritarar lendi í klípu eða geri villur þegar þeir reyna að breyta eða nota kóðann?
- ☐ Eru viðeigandi prófanir til staðar? Eininga, samþættinga, viðmóts eða end-to-end próf eftir því sem við á.
- ☐ Eru nöfn á hlutum viðeigandi og nægilega lýsandi?
- ☐ Eru góðar athugasemdir (e. comments) í kóðanum? Góðar ahugasemdir lýsa „af hverju" kóðinn er að gera það sem hann er að gera. Gæti einföldun á kóða orðið til þess að færri athugasemda er þörf?
- ☐ Er nýr kóði skrifaður í sama stíl og þegar er í lausninni (ekki verið að meina linting level hérna).
- ☐ Eru óþarfa persónulegar stílbreytingar í kóðanum sem tengjast ekki lausninni (t.a.m. skal ekki gera miklar breytingar á intendation eða bilum í ótengdum kóða eftir duttlungum hvers forritara).
- ☐ Er viðeigandi skjölun til staðar?
- ☐ Eru viðmótsbreytingar skynsamlegar og líta vel út? Samræmist útlit viðbóta því útliti sem er til staðar?
- ☐ Viðeigandi þolgæði (e. resilience) eru til staðar í kóðanum?

Viðauki 6 – Útgáfudagatal

Öll verkefni skulu fylgja útgáfudagatali sem skilgreinir þau tímabil sem útgáfur mega ekki vera gefnar út í production umhverfi, nefnt útgáfufrysting (e. deployment freeze, blackout periods). Þetta er nauðsynlegt til að forðast óþarfa útköll, vaktir og lágmarka truflun á fyrirsjáanlegum álagstímum fyrir kerfið.

Öll kerfi skulu útfæra sitt eigið útgáfudagatal sem inniheldur þær uppitíma kröfur eða útgáfufrystinga daga eða tímabil sem skipta máli fyrir viðkomandi kerfi.

Almennt

- Forðast skal útgáfur á föstudögum eða um helgar nema nauðsyn sé.
- Forðast skal útgáfur stuttu fyrir almenna frídaga eða löng frítímabil (þar sem fyrirsjánanlegt er að fámennt verður).
- Forðast skal útgáfur á sumarfrítíma.

Sniðmát

Eftirfarandi töflu má nota til hliðsjónar en hún inniheldur merkingar fyrir þau tímabil sem almennt ættu að vera skilgreind sem fasta frystingar ■ eða sem breytileg frysting ■ (t.d. kring um breytilega frídaga, s.s. páska).

Janúar	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Febrúar	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29		
Mars	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Apríl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Maí	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Júní	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Júlí	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Ágúst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
September	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Október	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Nóvember	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Desember	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31