



# Þróunarhandbók

Leiðarvísir fyrir

hugbúnaðarþróun fyrir

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 kítískt 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](mailto:mrh@landlaeknir.is)

## Efnisyfirlit

Áskorunin okkar .....	4
Viðhaldspolinn hugbúnaðararkitektúr.....	5
Skýr og einföld sjálfvirknivæðing .....	6
Kjarnyrnt skjölun.....	7
Höfum í huga að.....	7
Kóðageymslur.....	8
Skipulag kóðageymsla (e. code repository) .....	8
Skilastjórar .....	11
Hvernig ytri aðilar vinna með kóðageymslurnar .....	11
Nafnareglur á kóðagreinum.....	13
Gæða og útgáfufarlar.....	14
Skipulag útgáfunúmera.....	14
Sjálfvirkir gæðastýringarferlar .....	15
Hálf-sjálfvirkir gæðastýringarferlar.....	15
Sjálfvirkir útgáfufarlar .....	16
Á hvaða umhverfi eru útgáfur leyfðar á?.....	17
Sjálfvirkar gæða og öryggisprófanir .....	17
Skil á hugbúnaðarafurðum.....	20
Hverju á að skila? .....	20
Hver skilar? .....	20
Hvenær er æskilegt að skila inn? .....	20
Forritunarkóði .....	21
Forritunarmál og framework .....	21
Hönnun og arkitektúr.....	23
Forritunarkóði.....	24
Lagskipting og skipulag lausna .....	25
Prófanir .....	28
Endurnýting kóða / forritunarsöfn.....	29
Hugbúnaðarprófanir .....	31
Almennar uppsetningakröfur .....	31

Prófanir .....	32
Handvirkar prófanir .....	34
Samþykktarprófanir .....	34
Gögn sem þarf að afhenta með kóðanum .....	35
Aðgengismál .....	36
WCAG staðallinn .....	36
WAI-ARIA staðallinn .....	36
Önnur atriði tengd aðgengi .....	36
Hugbúnaðarskjölun .....	37
Hvaða skjölun er skilað við afhendingu .....	37
Hvaða skjölun verður að vera í GitHub .....	38
Viðkvæm gögn .....	40
Geymsla og miðlun gagna .....	41
Meðhöndlun heilbrigðisgagna .....	41
Gagnagrunnar .....	41
Dulkóðun gagna .....	42
Útgáfustjórnun .....	43
Útgáfuferli og keyrslustýring .....	44
Almennt .....	44
Docker .....	44
Kubernetes .....	46
Rekstur og eftirlit .....	47
Mælingar .....	47
Keyrslustaða .....	47
Útgáfuupplýsingar .....	48
Logging (upplýsinga og villuskráning) .....	49
Viðauki 1 – Sjálfvirkir útgáfuferlar í Github .....	50
Útgáfuferill heildarmynd .....	51
Útgáfuferill í skrefum .....	52
Viðauki 2 – Geymsla leyndarmála .....	53
Viðauki 3 – Samspil kóðagreina og skila .....	54
Viðauki 4 – Gátlistar .....	55

# 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 tæknistafla standist til lengri tíma litið.

Það sem við leggjum sérstaklega áherslu á er

- Framtíðartryggðar tækniákvæðanir
- Viðhaldspólinn hugbúnaðararkitektúr
- Skýr og einföld sjálfvirknivæðing
- Kjarnyrt skjölun

### Framtíðartryggðar tækniákvæðanir

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 tækni- og hverfi 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?

## Viðhaldspolinn 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

1. Móttöku og afhendingu gagna (e. request/response).
2. Stýringar (e. controllers).
3. Flæðistjórnun innan kerfisins.
4. Gagnasókn og gagnaskrif.
5. 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: Sannreyingar 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.

## 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 miðlægar 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.

## Nafnareglur á kóðageymslum

Kóðageymslur skulu fylgja eftirfarandi nafnareglum

1. Nöfn skulu ekki innihalda hástafi og nota bandstrik í stað bila eða undirstrika
  - a. `audit-ui-web`
2. Allar kóðageymslur sem tilheyra sömu lausn skulu bera sama forskeytið. Í eftirfarandi dæmi er forskeytið „audit-“
  - a. `audit-ui-web`
  - b. `audit-api`
  - c. `audit-service`



- d. audit-database
- e. audit-ui-app-android
- f. audit-ui-app-ios
- g. audit-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
<b>-SERVICE</b>	Fyrir bakendapjónustur sem eru <u>ekki aðgengilegar</u> utan eldveggs
<b>-API</b>	Fyrir bakendapjónustur sem eru aðgengilegar af internetinu (utan eldveggs)
<b>-DATABASE</b>	Gagnagrunnskóði
<b>-DATABASE-APEX</b>	Gagnagrunnskóði útfærður fyrir lausnir svipað og Oracle APEX.
<b>-DATABASE-PIPELINE</b>	Gagnagrunnskóði og annar kóði tengdur gagna meðhöndlunar pípum
<b>-INFRA</b>	Innviðakóði fyrir Kubernetes, Argo CD eða álíka (e. infrastructure as code)
<b>-INFRA-ARGO-PPT</b>	Innviðakóði ætlaður fyrir Argo CD á PreProduction umhverfi
<b>-INFRA-K8S-PROD</b>	Innviðakóði ætlaður fyrir Kubernetes á Production umhverfi

<b>-PACKAGE-NUGET</b>	Pakkar fyrir .NET Nuget kerfið
<b>-PACKAGE-NPM</b>	Pakkar fyrir Javascript NPM kerfið
<b>-PACKAGE-MAVEN</b>	Pakkar fyrir Java Maven kerfið
<b>-JOBS</b>	Forrit sem keyrð
<b>-DOCS</b>	Skjölun
<b>-APP-WEB</b>	Viðmótslausnir fyrir vef.
<b>-APP-ADMIN-WEB</b>	Viðmótslausnir fyrir stjórnenda vef
<b>-APP-UI-ANDROID</b>	Viðmótsforrit smíðað fyrir Android
<b>-APP-UI-IOS</b>	Viðmótsforrit smíðað fyrir Apple IOS
<b>-APP-API</b>	Bakendi sem einungis er ætlaður fyrir ákveðið viðmótsforrit. Ekki samnýttur með neinum öðrum forritum eða kerfum.
<b>-APP-CLI</b>	Console forrit, tekur við stýriskipunum úr console glugga en hefur ekki hefðbundið notendaviðmót.
<b>-APP-DOCS-ANDROID</b>	Skjölun fyrir viðmótsforrit smíðað fyrir Android
<b>-APP-DOCS-IOS</b>	Skjölun fyrir viðmótsforrit smíðað fyrir IOS
<b>-AUTOMATION-ACTIONS</b>	Github Actions sjálfvirknikóði skrifaður fyrir pípulínur
<b>-AUTOMATION-PA</b>	Sjálfvirknikóði skrifaður fyrir Microsoft Power Automate
<b>-DASHBOARD-POWERBI</b>	Mælaborð skrifað í Microsoft PowerBI
<b>-DASHBOARD-WEB</b>	Mælaborð skrifað sem vef sérsníði
<b>-CONFIG</b>	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 sjálfvirkir fjarlægðir af kóðageymslum. Tilkynna ber til okkar ef verktaki skiptir um skilastjóra.

## Hvernig ytri aðilar vinna með kóðageymslurnar

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

Við gerir 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.

Fork-and-Branch ferlið virkar í stórum dráttum á eftirfarandi hátt:

1. Verktaki býr til **fork** af repoinu í GitHub Landlæknis og vistar yfir á GitHub svæði verktaka.
  - a. Ath, hvort búa þarf til fork af **main** eða ákveðnu útgáfu tagi
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 forkinn í þínu GitHub
8. Opna PR frá útgáfubrančinu yfir í Repoíð hjá Landlækni
  - a. Athugið að öll commit skulu vera **squashed** og einungis eitt merge commit er leyft til landlæknis
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, töggðu og sameinuðu við Repoíð hjá Landlækni, þá þarftu að uppfæra forkinn í þínu GitHub
  - a. `git pull upstream main`
  - b. `git push origin main`

Meðfylgjandi er nánari yfirferð yfir helstu þætti þessarar aðferðarfræði

<https://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/>

Nánari upplýsingar er að finna í Viðauka 3.

## Nafnareglur á kóðagreinum

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

Titlar/nöfn á breytingabeiðnum þurfa að vera nákvæmlega eins og nafn á viðkomandi kóðagrein.

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.
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.
test/2024.08.05-dev	Prófunarútgáfur af hugbúnaði sem ætlaður er til sjálfvirkrar útgáfu á test umhverfi.

# 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.

### 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

Allir sjálfvirkir ferlar eru keyrðir af GitHub Actions, nánari lýsingar er að finna í Viðauka 1.

Fyrir hver kóðaskil, PR (e. pull request), 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 kóðarýni 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 höfundur 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.

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.

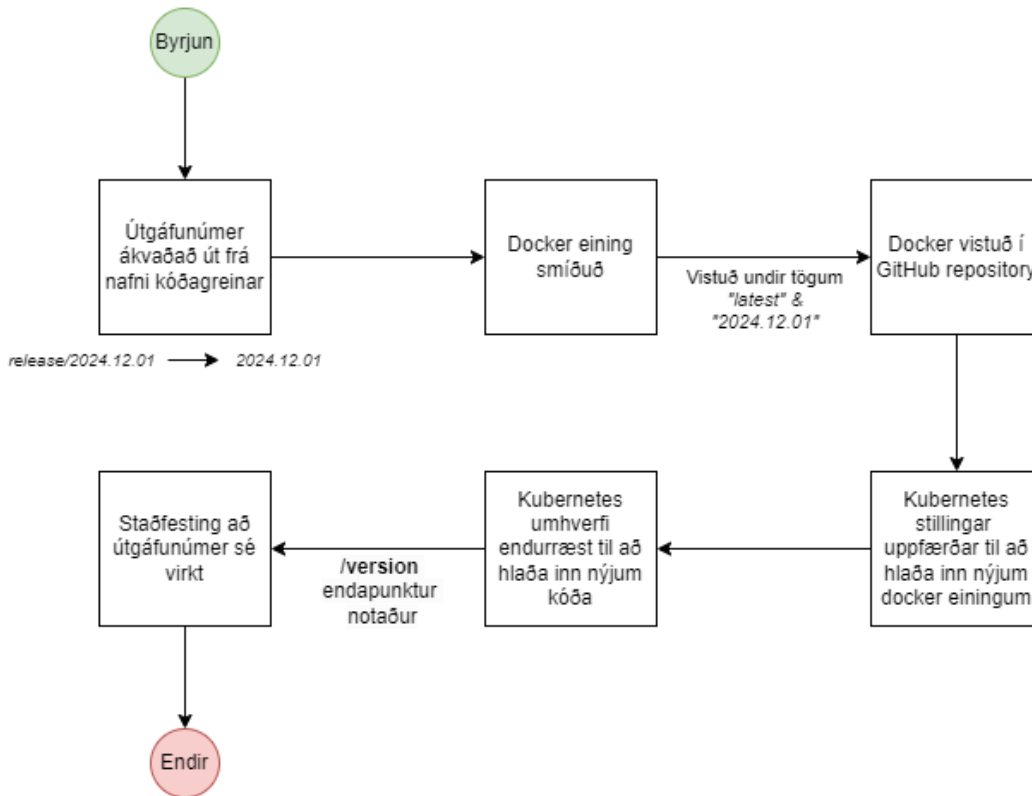
## Sjálfvirkir útgáfufarar

Allur hugbúnaður sem ætlaður er til útgáfu á rekstrarumhverfi 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 útgáfufarla og til þess að draga úr líkum á mannlegum mistökum við útgáfu hugbúnaðar.

Sjálfvirk útgáfufarar 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

## Útgáfufarillinn í mynd





## Á hvaða umhverfi eru útgáfur leyfðar á?

Sjálfvirkir útgáfufarar 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	<i>latest</i>	Sjálfvirk/Handvirk	Ekki leyft	Ekki leyft
Pre-Production	<i>latest-qa</i>	Sjálfvirk	Sjálfvirk	Ekki leyft
Test	<i>latest-test</i>	Handvirk	Handvirk	Sjálfvirk

## 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

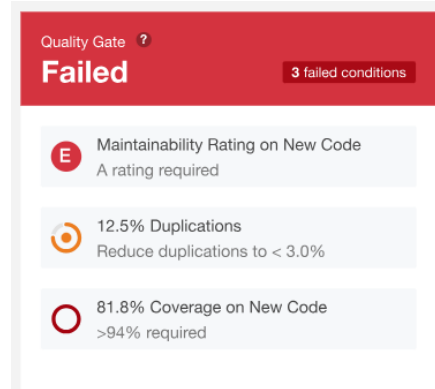
- C# .csproj skrár (eða Directory.Build.props)
  - CodeAnalysisRuleSet, GenerateDocumentationFile, tög séu til staðar
  - TreatWarningsAsErrors sé til staðar og sett á „true“
  - Að eftirfarandi pakkar séu til staðar
    - StyleCop.Analyzers

- 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

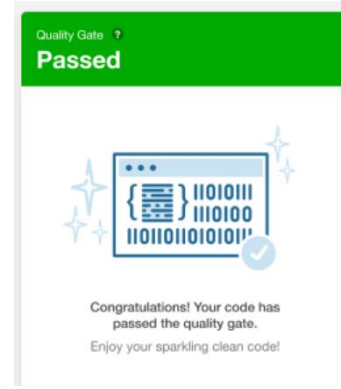
### Sjálfvirk gæðahlið (e. Quality Gates)

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 fyrirbyggjandi vandamál í eldri kóðahlutum sem eru ótengd þeim breytingum sem verið var að gera (e. clean-as-you-code aðferðarfræði).



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 fyrir 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.

## Hvenær er æskilegt að skila inn?

Skil	1x í viku	>=1x í mánuði	<1x í mánuði	Í lok verkefnis
Á test umhverfi	Æskilegt	Í lagi	Óæskilegt	Óæskilegt
Á pre-production umhverfi	Í lagi	Æskilegt	Æskilegt	Óæskilegt
Á production umhverfi	Óæskilegt	Óæskilegt	Í lagi	Æskilegt

# 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 keyrsuumhverfi 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. Notast skal við þau framenda framework sem eru þegar ráðandi í lausnum hjá okkur. Ekki skal smíða lausn í forritunarmáli eða umhverfi sem ekki hefur verið notað áður nema með leyfi sérfræðinga okkar.

Allir framendur 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, SSR) fram yfir virkni sem keyrir eingöngu í vafra notanda. Dæmi um slíka tækni er [Blazor.NET](#) og [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 notast við SSR virkni í SPA forritunarsöfnum (líkt og Angular, React, nextjs, Svelte, etc).

Ef SSR nálgun er ekki fýsileg og þörf er á að útfæra SPA vefsíðu sem keyrir í vafra notenda þá skal stuðst 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 undir neinum kringumstæðum** útfæra veflausnir fyrir okkur í vefumsjónarkerfum án skriflegs samþykkis frá 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.

## Viðmót og vefir

Leitast skal við að hönnun og forritun á vef og app framendum fylgi micro-frontend hönnunarmynstrum 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 lóðréttu sneiðar (e. vertical slice architecture) frekar en láréttar lagskiptingar þar sem hentar. Hvatt er til að skoðuð sé REPR<sup>1</sup> (e. request endpoint response) högun fyrir hefðbundna WebApi högun frekar en MVC haganir (e. model-view/viewmodel-controller).

Allar þjónustur skulu hafa OpenAPI skilgreiningar 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 hvaða auðkenningarmáti er notaður í forritunarskilum þjónustunnar og skal vera hægt að prófa þjónustuna á sjálfvirkan máta óháð ytri auðkenningarþjónustum.

Aðgreina skal með viðskeyti á nafni á milli þjónusta sem eiga að vera aðgengilegar á internetinu og þeirra sem sitja bak við eldvegg. Nota skal -api viðskeyti fyrir þjónustur sem eru aðgengilegar internetinu,

---

<sup>1</sup> <https://www.apitemplatepack.com/docs/introduction/repr-pattern/>  
<https://deviq.com/design-patterns/repr-design-pattern>

viðskeytið -service skal nota fyrir þær sem sitja fyrir innan eldvegg. T.d. auditing-service og auditing-api. Sjá nánari umræðu um viðskeyti í kaflanum um Skipulag kóðageymsla.

## 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 regluettunum 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. 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.

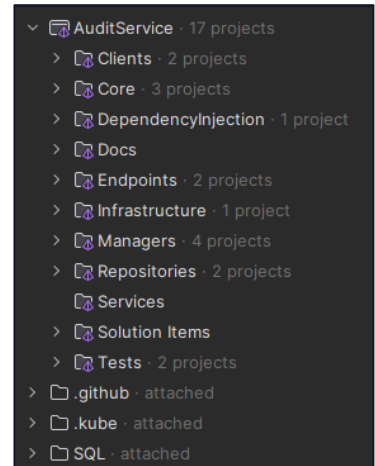


## Lagskipting og skipulag lausna

Hver lausn þarf að fylgja eftirfarandi rótarmöppu (e. top level) uppskiptingu fyrir forritunarkóða. Skiptingunni er ætlað að framfylgja onion lagskiptinguna. 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. í 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.

## 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 þökkum (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 rötarmöppuna Tests/UnitTests/
- Uppbygging einingaprófanna skulu endurspegla ytra skipulag lausnarinnar (url path), en hafa rötarmöppuna 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:

build summary

### Test Results

70 tests	+1	70	✓ +1	2s	🕒 ±0s
1 suites	±0	0	⚡ ±0		
1 files	±0	0	✗ ±0		

Results for commit a414ea1a. ± Comparison against earlier commit 979bc441.

Job summary generated at run-time

- 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 heimasíðað eða óreynt framework til einingaprófana sem hluta af hugbúnaðarlausninni né nota eitthvað sem var smíðað af verktaka og ætlað fyrir önnur verkefni

### 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 fyrirbyggjandi 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>

# 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

### Ut anumhald 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 skírt 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 sjálfvirk
- 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);
}

```

## Sambættingarprófanir (integration tests)

- Öll föll í ytri þjónustuskilum skulu vera prófuð af a.m.k. einu sambættingaprófi sem staðfestir virkni aðgerðar (e. success criteria) og öðru sambættingaprófi sem staðfestir villumeðhöndlun aðgerðar (e. failure criteria)
- Halda skal sambættingarprófunum í lágmarki og einblína á að prófa virkni og réttleika sambættingaskila í kerfinu (sbr. samskipti við aðra APIs, gagnagrunna eða kerfi).
- Sambættingapróf 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 sambættingapróf.

## 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 fækjustigi viðmótsprófana í algjöru lágmarki og einblína á að prófa einungis hluta sem eru krítískir fyrir virkni kerfisins.

- Nota skal Cypress, <https://www.cypress.io/>, við smíði viðmótsprófana nema eðli verkefnisins krefjist annars.

### Aðgengisprófanir (e. accessibility 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 tilfalli 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 inni 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 | - 2.2.3 | - 2.4.8  |
| - 1.4.7 | - 2.2.5 | - 2.4.10 |
| - 1.4.8 | - 2.2.6 | - 2.4.12 |
| - 1.4.9 | - 2.3.2 | - 3.3.6  |
| - 2.1.3 | - 2.3.3 | - 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. 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 skrá (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 gagnafæði 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”: hagrinn af valinu, hvernig valið uppfyllti skilyrðin sem lögð voru til grundvallar
- “vitandi það að”: lýsing á veikleikum eða takmörkunum sem eru þekktar tengdar ákvörðuninni, 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), innviðabarfir, 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.*

- 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 8 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 hvernar 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
- **continuous-integration.md** skrá í rót verkefnis eða repo eins og það á við
  - Continuous integration/deployment, lýsingu á hvernig github actions eru settar upp, branching strategy fyrir repoð (dev work, og svo útgáfu strategía), container repo fyrirkomulag
  - Útgáfunúmerastýring, útgáfu fyrirkomulag úr github,
- **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 skal miðlað á öruggan hátt til ábyrgðaraðila verkefnisins.

- Dulritunarlyklar, öryggisskilríki,
- Upplýsingar um allar aðgangstillingar fyrir tengdar þjónustur (s.s. username, passwords, client\_id, api\_keys etc)



# 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 líkt og Signet.
- Ekki deila eða birta gögn í videó fundum, t.d. teams samtölum.
- 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). Ef nauðsynlegt er að velja aðra tegund gagnagrunns þarf að fá leyfi fyrir því frá sérfræðingum og gæðastjóra.

### Forritunarpakkar (e. ORM)

Hugbúnaðarteymi sem eiga samskipti við gagnagrunna skulu velja léttu gagnagrunns forritunarpakka (e. ORM, object-relational mapping).

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 ORM ásamt uppfærslu (e. migration) stefnu sem myndar ekki óhóflegt magn af viðbótar gagnagrunnshlutum, sem gera erfitt að vinna með gagnagrunnskemað í gagnagrunninum (t.d. töflunöfn innihaldi GUID gildi og þannig háttar).

### 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 lykklum milli gagnagrunnskerfa og uppfærsla.

### Dulkóðun gagna

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.

Ef dulkóða þarf gögn í einstaka svæðum í töflu, (t.a.m. kennitölur eða sjúkdómsgreiningar) þá skal notast við dulkóðunarlykil embættisins. Þetta er sérstaklega 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 fyrirfram ákveðnu gildi við gögnin áður en þau eru hash'uð. 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 dulrita í gagnagrunnstöflur og geyma.

## Útgáfustjórnun

Geyma skal og útgáfustýra skal öllum gagnagrunnskóða í GitHub repoi MRH. Ef hægt er skal geyma gagnagrunnskóðann sem SQL scriptur í sér repo (sjá **Hvernig ytri aðilar vinna með kóðageymslurnar**), ef slíkt er ekki hægt eða óskynsamlegt skal geyma SQL scripturnar með þjónustukóða bakendaþjónusta í rótarmöppu sem heitir */SQL*.

# 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 samtali 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 samtali snemma í þróunarferlinu (helst áður en forritun hefst).

### 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
- **Build Process**
  - Skal vera eins einfalt og mögulegt er
  - Samræmt milli umhverfa (Dev, Pre-Production, Production)
  - Á **EKKI** að byggja forritunarkóða (e. 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 sem lýst er í Skipulag útgáfunúmera. Athugið að tögum er úthlutað sjálfvirkt í pípulínunum
- 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 skráum 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 skrána
- **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 má ekki**
  - 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 keyrslumhverfi lausnarinnar skal geyma í docker. Keyrsluleyndarmál skal geyma í Vault og senda inn í docker eininguna 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 skrá) m)
- 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 endapunktur)
- Allar þjónustur skulu prenta strúktúraðar logfærslur beint út í console out, ekki skal skrifa logga í skrár á disk
- Öll keyrsluleyndarmál (s.s. gagnagrunnstengingar, aðgangsslyklar, lykilorð og notendanöfn) skulu geymd í Vault. Leyndarmálum skal skotið inn í docker containera rétt áður en þeir eru keyrðir upp

# 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 sjálfvirknivæðingu. Af þessum sökum er mikilvægt að lausnir safni saman og skili af sér á staðlaðan hátt mæligögnum, keyrslustöðu og villuskram sem geri bilanaleit og rekstrarvöktun bæði skilvirka og einsleita. Einnig er mikilvægt að kerfisstjórar og rekstraraðilar hafi góðan skilning á innviða og tengipörfum hugbúnaðarins.

### 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 mun því takmarka 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.

*Mælingarendapunktur ættu að vera aðgengilegir á öðru porti (9102) en aðrir almennir endapunktur til að auðveldara sé að loka á utanaðkomandi aðgang að þeim í eldveggjum.*

### Keyrslustaða

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

- /system/alive

- Er þjónustan 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
  - Er þjónustan 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 útgáfufærla 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).

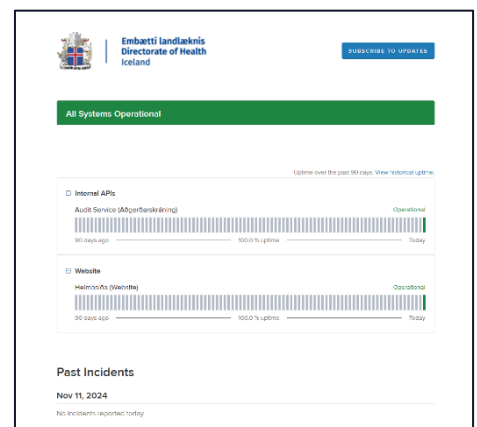
### Viðbótar svæði

ECS Schema path	Lýsing
<b>labels.x-road-client</b>	Skráning á x-road upplýsingum sem koma inn með x-road-client http header.
<b>labels.x-correlation-id</b>	Skráning á rakningarupplýsingum sem sendar eru inn með x-correlation-id http header.
<b>labels.application</b>	Nafn hugbúnaðarins (þetta þarf að vera einkvæmt nafn). Miðað 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 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.

Útgáfuferlum er stýrt af Github Actions.

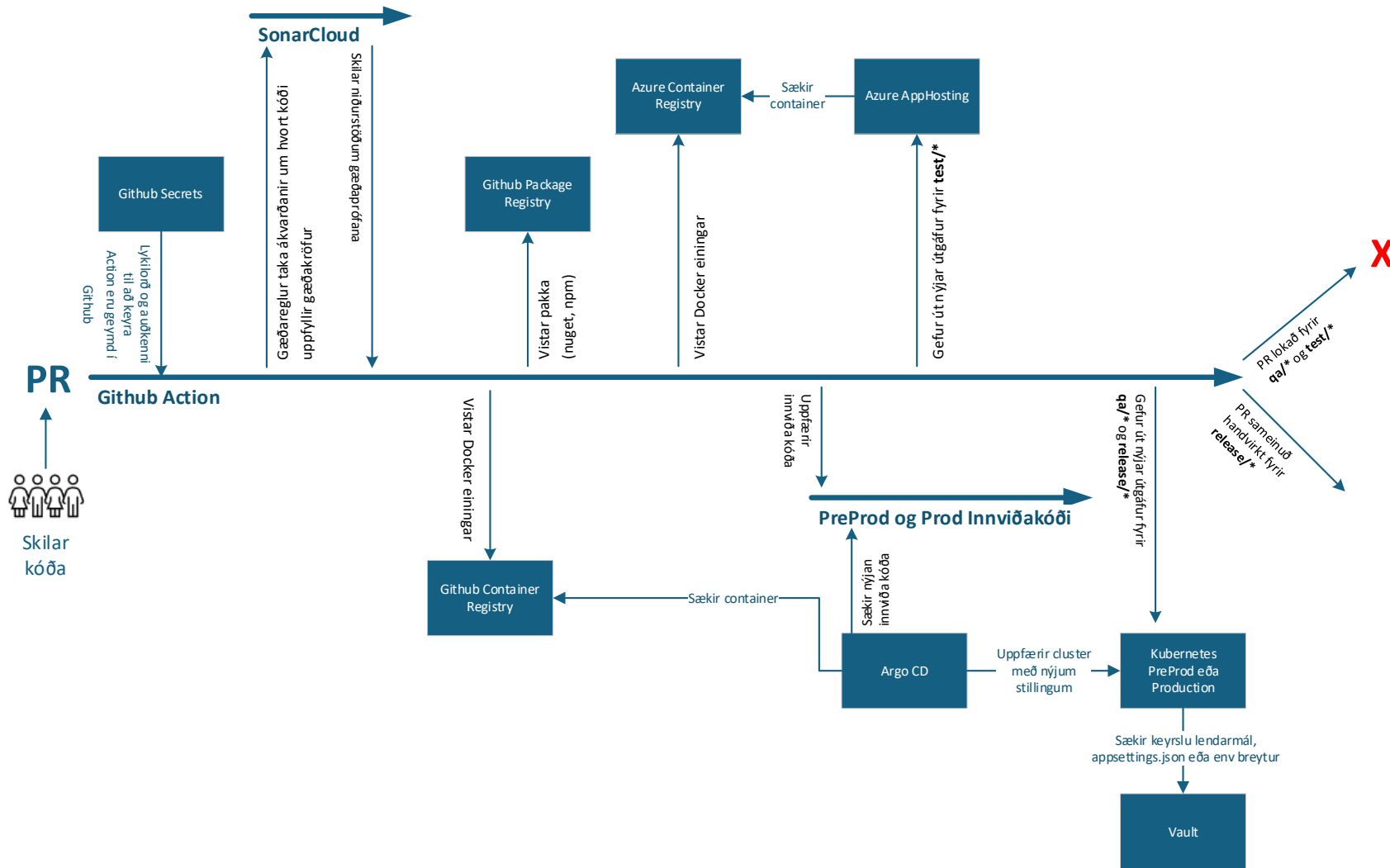
Við gefum út og viðhöldum sniðmát af útgáfuferlum og er það sniðmát aðgengilegt öllum þegar nýjar kóðageymslur eru búnar til.

Mikilvæg atriði sem ber að hafa í huga

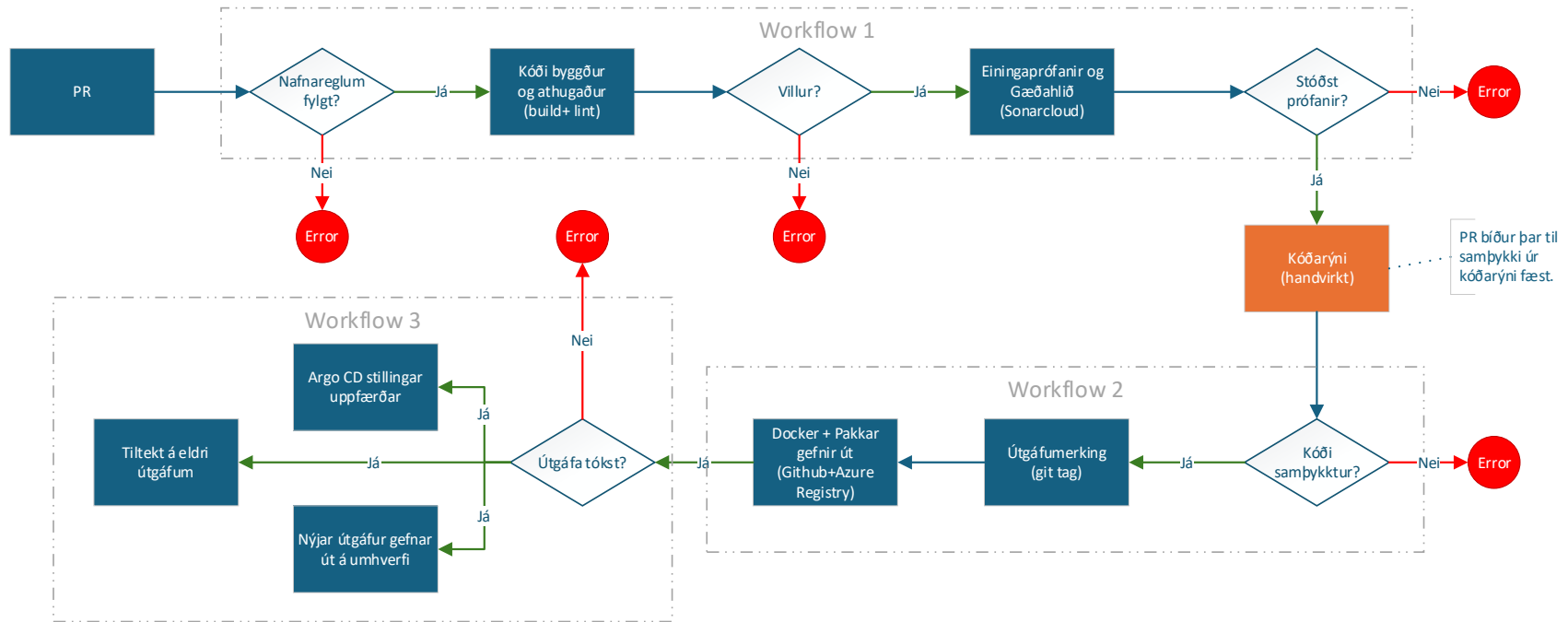
- Allir innviðir eiga að vera geymdir sem kóði (e. infrastructure as code). Bæði sem docker skipanir og sem kubernetes skilgreiningar.
- Github Actions píplínur eiga að sjá um að útbúa kóða fyrir útgáfu (build, compile, test). Leitast skal við að halda Docker skilgreiningum eins einföldum og mögulegt 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 öll keyrsluumhverfi. Þ.e.a.s. „latest“ er tag sem gefið er út á Production, „latest-qa“ er gefið út á PreProduction og „latest-dev“ er tagið sem gefið er út fyrir Test.
  - Vegna þessa 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.

# Útgáfuferrill heildarmynd

Eftirfarandi útgáfuferrill



## 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.

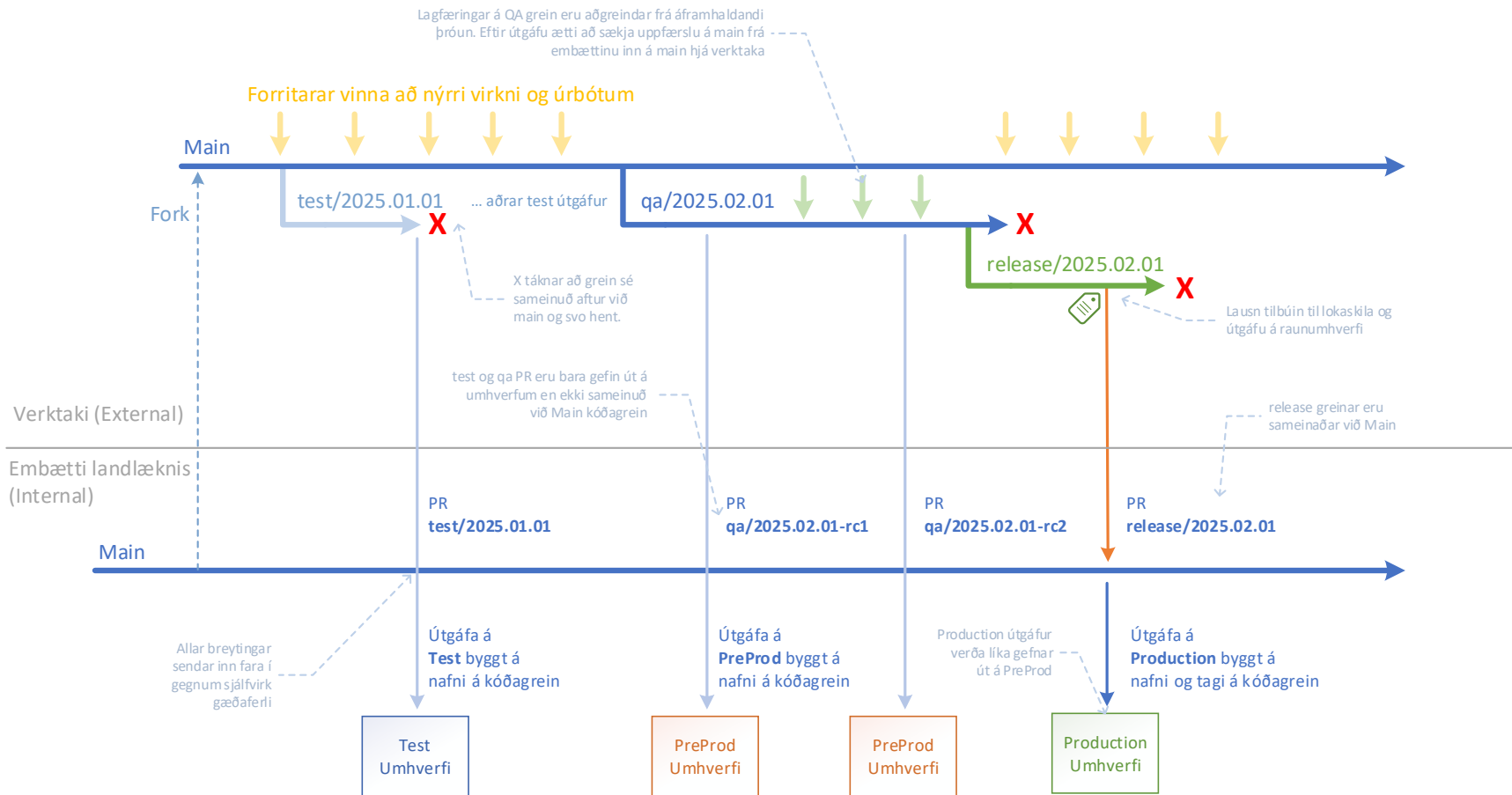
Í innviðakóða skal notast við Hashicorp Vault þjónustu sem sett er upp í hverju Kubernetes clusteri fyrir sig.

Ath, ekki skal geyma gögn sem ekki teljast leyndarmál í secret geymslum. Venjulegar Environment breytur skal geymast í kubernetes skráum fyrir viðkomandi umhverfi.

Leyndarmál skal geyma annað hvort sem heil JSON skjöl (t.d. í .NET, appsettings.json) eða sem Environment breytur (t.d. í vefkóða).

Ef notast er við environment breytur skal kóðinn ekki nota mismunandi nöfn fyrir breytur eftir umhverfum. T.d. skal ekki nota API\_URL\_PROD og API\_URL\_PPT heldur einungis API\_URL.

# Viðauki 3 – Samspil kóðagreina og skila



# Viðauki 4 – Gátlistar

## Gátlisti áður en forritun hefst

- 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
  - Ákveðið hvar og hver eigi að geyma gögnin (hvaða gagnagrunnur).
  - Gagnagrunnsskema (ef við á) er samþykkt.
  - Aðgangur veittur að gagnagrunnskerfi
  - Kóða varðveisla í Github fyrir 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 aðlagðar 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 endapunktur 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](https://landlaeknir.statuspage.io)
- Útgáfuferli fyrir umhverfi
  - Test
  - PreProd
  - Production