4. ÚTGÁFA | JÚLÍ 2026

Þróunarhandbók Viðaukar

Viðaukar við leiðarvísi fyrir þróun stafrænna lausna fyrir íslenskar heilbrigðisstofnanir

Stjórnarráð Íslands
Heilbrigðisráðuneytið | Stafræn Heilsa

Útgefandi:
Heilbrigðisráðuneytið | Stafræn Heilsa

Þróunarhandbók Viðaukar

Viðaukar þessir eru skrifaðir af Sverri Sigmundarsyni, Einari Loga Einarssyni og Ólafi Kristjáni Ragnarsyni hjá Stafrænni Heilsu.

4. ÚTGÁFA | JÚLÍ 2026

sh@hrn.is

https://stafraenheilsa.is

Umbrot og textavinnsla:
Heilbrigðisráðuneytið

ISNB 978-9935-589-00-2

Þessi þróunarhandbók skilgreinir kröfur, verklag og leiðbeiningar sem snúa að þróun, gæðaeftirliti og útgáfu hugbúnaðar fyrir verkkaupa.

Handbókin er hluti af heildstæðu handbókakerfi og skal lesa hana í samhengi við:

- Tæknistefnu verkkaupa
- Tæknikröfur verkkaupa
- Rekstrarhandbók verkkaupa
- Hönnunarhandbók verkkaupa

Ef árekstur verður milli skjala gilda tæknistefna og tæknikröfur um tæknilegt val og tæknilegar afstöður. Um þróunarferla, gæðakröfur og skil gildir þróunarhandbók þessi.

Allir sem lesa þennan texta eru hvattir til að hugsa gagnrýnið um efni skjalsins og koma með tillögur að úrbótum til verkefnastjóra.

Efnisyfirlit

1. Fyrirvari

Þróunarhandbók þessi er eign verkkaupa og er frekari birting óheimil nema með samþykki.

Kröfur sem settar eru fram í þessari handbók eru bindandi í útboðum, samningum og verkefnum þar sem handbókin er hluti af samningsgögnum. Frávik frá bindandi kröfum eru einungis heimil í samræmi við skilgreindan undanþáguferil (*sjá viðauka Undanþáguferill frá stefnu verkkaupa*).

Lausnir í lausnabanka verkkaupa eru birtar „eins og þær koma fyrir“ (as-is), án ábyrgðar á réttleika eða hæfi til notkunar í tilteknu verkefni. Ekki er ætlast til að kóði úr lausnarbanka sé notaður óbreyttur eða órýndur í lausnir verksala**,** sama gildir um fyrir fram tilbúnar forritunarlausnir sem verksali leggur til verksins. Allur kóði sem er hluti af skilum fellur undir sömu kröfur og annar hugbúnaðarkóði, óháð uppruna.

Hvers kyns notkun á lausnum eða tillögum úr þessari handbók eða lausnarbanka, eins og þær koma fyrir, er á ábyrgð verksala og verkkaupa að skaðlausu.

2. Breytingar frá fyrri útgáfu

Nýtt efni

- Viðaukaskjal stofnað sem sjálfstætt skjal, aðskilið frá meginskjali.
- Viðauki — Skipulag. agent/ möppu: Agent OS v3.0, möppuskipulag, staðlaskrár og kröfur til verksala.
- Viðauki — Gervigreindarlausnir — arkitektúr og kröfur: Lagskipting agenta, MCP-högun, öryggiskröfur, prófanir og viðmiðatafla.
- Viðauki — Kóðastíll og gæðatól — verkfæri, stillingar og reglusett: StyleCop, Roslyn Analyzers, þýðistillingar, gæðahlið viðmið.
- Viðauki — Lagskipting og möppuskipan lausna: Rótarmöppuskipan, meginreglur lagskiptingar, óheimil hugtök.
- Viðauki — Kröfur til OpenAPI skilgreininga: Lágmarkskröfur, gæðakröfur, gagnasnið.
- Viðauki — Skipulag og snið hugbúnaðarskjölunar: /Docs möppuskipulag, README kröfur, CronJob skjölun, gæðamat.
- Viðauki — Docker kröfur og bestu venjur: Build ferli, öryggi, stærð, logging, merking.
- Viðauki — Kubernetes kröfur og samhæfni: GitOps, Argo CD, heilsuathuganir, stillingar.
- Viðauki — Samspil kóðagreina og skila.
- Viðauki — Yfirferðarskjal kóðarýni flutt í viðauka.

Breytingar

- Viðauki — Útgáfuferlar (QA og PROD): Endurskrifaður og rýmkaður, nú með skýrri ábyrgðarskiptingu.
- Viðauki — Sjálfvirkir útgáfuferlar í GitHub: Endurskrifaður til samræmis við nýtt CI/CD-fyrirkomulag.
- Viðauki — Stöðluð forritunarstöfn verkkaupa: Uppfærð og rýmkuð. Söfn auðkenningar og DI bætt við.
- Viðauki — Geymsla leyndarmála: Endurskrifaður — nú nær til GitHub Actions, Kubernetes/Vault,.NET appsettings og legacy lausna.
- Viðauki — Útgáfunúmer og merking: Endurskrifaður.
- Viðauki — Skipulag prófana og tæknilegar útfærslur: Rýmkaður, nú með prófunarrömmum og undanskildum kóða.

Fjarlægt efni

- Útgáfudagatal (var í útgáfu 3) — fjarlægt þar sem það er ekki hluti af þróunarhandbók.

3. Viðauki – Dæmi um nafnagjöf kóðageymsla

Sérfræðingar SH stofna allar nýjar kóðageymslur í GitHub-umhverfi verkkaupa við upphaf verkefna.

Þegar send er inn beiðni þarf nýja kóðageymslan að innihalda nafn yfirverkefnis og svo nafn virkninnar sem búa á til. Til dæmis forritari er að vinna að lausn fyrir bókunarkerfi fyrir heilsuveru, hann þarf þá 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“

4. Viðauki – Útgáfuferlar (QA og PROD)

Þessum ferlum er ætlað að

- tryggja að verktakar geti sinnt þróunarvinnu í sínu eigin umhverfi og samkvæmt eigin vinnureglum
- viðhalda skýrri ábyrgðarskiptingu milli verktaka og verkkaupa
- tryggja rekjanleika, gæði og öryggi hugbúnaðarafurða
- tryggja að einungis samþykktur hugbúnaður fari í prófunar- og rekstrarumhverfi verkkaupa

4.1 Upphaf og ábyrgð

Verkkaupi stofnar og heldur utan um allar kóðageymslur í GitHub-umhverfi sínu. Við upphaf verkefna sér verkkaupi um að:

- setja upp sjálfvirka byggingar- og útgáfuferla (CI/CD), þar á meðal GitHub Workflows og tengdar pípulínur
- skilgreina CODEOWNERS og aðgangsstýringar
- útbúa grunnskrár og skilgreindan möppustrúktúr fyrir lausn
- veita viðeigandi verktakateymum aðgang að kóðageymslum samkvæmt skilgreindum teymum í GitHub

4.2 Þróunar- og skilferli (fork og Pull Requests)

1. Verktaki forkar kóðageymslu verkkaupa í sitt eigið GitHub-umhverfi.
2. Þróun fer fram í „fork“ verktaka, í samræmi við hans eigin vinnureglur, verkfæri og öryggiskröfur.
3. Þegar breytingar eru tilbúnar til skila til verkkaupa stofnar verktaki viðeigandi kóðagrein (t.d. feature/* eða hotfix/*) í forki sínu.
4. Verktaki sendir breytingarnar til verkkaupa með breytingabeiðni (Pull Request).
5. Verkkaupi framkvæmir yfirferð á breytingabeiðni, þar á meðal:
    - kóðarýni
    - keyrslu skilgreindra sjálfvirkra prófana
    - keyrslu gæða- og öryggisferla (t.d. statíska greiningu)
6. Að lokinni samþykkt er breytingum sameinað í main-grein kóðageymslu verkkaupa.

4.3 Almennar reglur um útgáfur

1. Allar útgáfur hugbúnaðar eru byggðar á samþykktum kóða í main.
2. Ekki er heimilt að gefa út hugbúnað beint af öðrum kóðagreinum.
3. Hotfix-útgáfur skulu ávallt sameinaðar aftur inn í main að lokinni útgáfu.
4. Útgáfur eru skilgreindar með merkingum (tags) í Git og eru einungis slíkar merktar útgáfur heimilar til uppsetningar í prófunar- og rekstrarumhverfum.

4.4 Ferli til útgáfu í prófunarumhverfi (QA / Pre-production)

1. QA-útgáfa er skilgreind með merkingu (tag) sem byggir á samþykktum kóða í main.
2. Engar breytingar eru heimilar á QA-útgáfu eftir að hún hefur verið skilgreind.
3. Sjálfvirkir byggingar- og prófunarferlar eru keyrðir.
4. Að lokinni sjálfvirkri yfirferð er QA-útgáfa samþykkt án handvirkrar íhlutunar.
5. Keyrslueiningar eru byggðar og merktar samkvæmt skilgreindum reglum.
6. Sjálfvirk útgáfa fer fram í prófunarumhverfi.

4.5 Ferli til útgáfu í rekstrarumhverfi (PROD)

1. Rekstrarútgáfa er skilgreind út frá áður samþykktri QA-útgáfu eða hotfix-merkingu.
    1.1. Verksali þarf að tryggja að hægt sé að sameina breytingar í hotfix inn í main eftir útgáfu.
2. Engar breytingar eru heimilar á rekstrarútgáfu eftir að hún hefur verið skilgreind.
3. Sjálfvirkir byggingar- og prófunarferlar eru keyrðir.
4. Útgáfa í rekstrarumhverfi krefst handvirkrar samþykktar af hálfu verkkaupa (tækniteymis eða gæðastjóra).

5. Að lokinni samþykkt eru keyrslueiningar byggðar og merktar.
6. Sjálfvirk útgáfa fer fram í rekstrarumhverfi.

5. Viðauki – Rekjanleiki milli kóðagreina, breytingabeiðna og útgáfa

Titlar/nöfn á breytingabeiðnum 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“.

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.

Kóðagrein (e. branch name)	**Lýsing**
release/2024.08.05 release/2024.08.05-hotfix1	Loka útgáfur af fyrir fram ákveðnum útgáfum eða neyðarútgáfum af hugbúnaði sem ætlaður er til útgáfu á pre-production og production umhverfum. Athugið að 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).
feature/HEITI_Á_VIRKNI	Ný virkni, notuð til móttöku kóða.
hotfix/2024.08.05-LÝSING	Bráðabirgðalagfæringar á kóða sem er í keyrsluumhverfi.

6. Viðauki – Sjálfvirkir útgáfuferlar í Github

Markmið sjálfvirknivæðingar í útgáfuferlum verkkaupa er að:

- lágmarka möguleika á mannlegum mistökum,
- tryggja að leyndarmál séu varðveitt og miðlað á öruggan hátt,
- tryggja að ferlar sem nauðsynlegir eru til útgáfu séu skýrt skjalaðir og endurtakanlegir.

6.1 Keyrsluumhverfi

Allar lausnir verkkaupa keyra í Docker-einingum í Kubernetes-umhverfum. Keyrsluumhverfin eru tvö:

Umhverfi	**Tilgangur**
Pre-Production (PreProd)	Samþættingar- og gæðaprófanir á útgáfum
Production (Prod)	Raunkerfi ætlað almenningi og heilbrigðisstarfsfólki

Bæði umhverfi eru hýst í Kubernetes-klösum í umsjón verkkaupa. Sjá viðauka „Þjónustur og verkfæri verkkaupa“ fyrir nánari upplýsingar um hýsingarveitendur.

Verkkaupi notar Argo CD til umsýslu og eftirlits á útgáfum og keyrsluumhverfi. Allar Argo CD og Kubernetes stillingar skulu geymdar í .kube/ möppu með lausnarkóðanum.

6.2 Sjálfvirkir gæðaferlar við kóðaskil

Fyrir hverja breytingabeiðni (Pull Request) sem berst kóðageymslum verkkaupa fer eftirfarandi sjálfvirkt ferli í gang:

Skref	Lýsing	Afleiðing ef bregst
1	Kóðinn prófaður m.t.t. uppbyggingar lausnar	PR hafnað
2	Kóðinn fer í gegnum build ferli	PR hafnað
3	Einingaprófanir keyrðar	PR hafnað
4	Sjálfvirk kóðarýni og kóðagæðaeftirlit keyrt	PR hafnað
5	Docker einingar fara í gegnum öryggisprófun	PR hafnað
6	Allur gagnagrunnskóði fer í gegnum gæðaeftirlit	PR hafnað

Ef eitthvert sjálfvirkt skref skilar villu er breytingabeiðni sjálfkrafa hafnað og verksala ber að lagfæra og endursenda.

Sjá kafla 6 (Gæðaferlar) og viðauka „Kóðastíll og gæðatól“ fyrir nákvæm viðmið og stillingar.

6.3 Fyrirkomulag CI sjálfvirknivæðingar

6.3.1 Innviðakóði

Innviðakóði er geymdur í Git-kóðageymslum samkvæmt GitOps-hugmyndafræði, undir .github/ möppu í hverri lausn.

6.3.2 CI/CD ferlar verkkaupa

CI/CD ferlum verkkaupa er stýrt af GitHub Actions. Verkflæðisskrár (workflows) verkkaupa eru í .github/workflows/ möppu hverrar lausnar.

Verkkaupi gefur út og viðheldur sniðmátum af GitHub-útgáfuferlum sem eru aðgengileg öllum þegar nýjar kóðageymslur eru stofnaðar.

6.3.3 CI/CD ferlar verksala

Verksölum er frjálst að nýta þau CI-sjálfvirkniverkfæri sem best henta þeirra vinnustíl í eigin fork. Mælt er með að umhverfin séu byggð á Git, t.d. GitHub Actions, GitLab CI, Bitbucket Pipelines eða Harness.

Ef verksali notar einnig GitHub Actions í fork sínum skal nota forskeytið xternal- á sínum verkflæðisskrám til að aðgreina frá skrám verkkaupa. Dæmi:

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

Ekki er þörf á forskeytum ef verksali notar aðrar CI-þjónustur (t.d. .gitlab-ci.yml eða bitbucket-pipelines.yml).

6.3.4 Aðgreining milli umhverfa

Verkflæðisskrár verkkaupa innihalda stýringar sem hindra að þær séu keyrðar í öðrum umhverfum en hjá verkkaupa. Ef verksali notar GitHub Actions er mælt með sambærilegum stýringum til að hindra keyrslu í umhverfi verkkaupa.

Dæmi:

```
jobs:
  job-not-for-buyer:
    if: github.repository_owner != '<ORG_NAFN_VERKKAUPA>
    steps: ...
```

Verkkaupi tilgreinir raunverulegt organization nafn við upphaf verkefnis.

6.4 Reglur um Docker-merkingar og útgáfur

Eftirfarandi reglur gilda um merkingar (tags) á Docker-einingum og útgáfunúmer:

Atriði	**Regla**
Merking á Docker-einingum	Bæði útgáfunúmeri og git commit SHA
PreProd merking	CalVer með -rc viðtengingu (t.d. 2025.12.09-rc1)
Production merking	CalVer er með eða án viðtengingar (t.d. 2025.12.09 eða 2025.12.09-hotfix1). -rc viðtenging er aldrei heimil í Production
Föst tög (t.d. latest)	Óheimil í PreProd og Production
imagePullPolicy	Kubernetes-poddar skulu ávallt vera stilltir með imagePullPolicy: Always

Sjá viðauka „Útgáfunúmer og merking“ fyrir fulla CalVer-skilgreiningu.

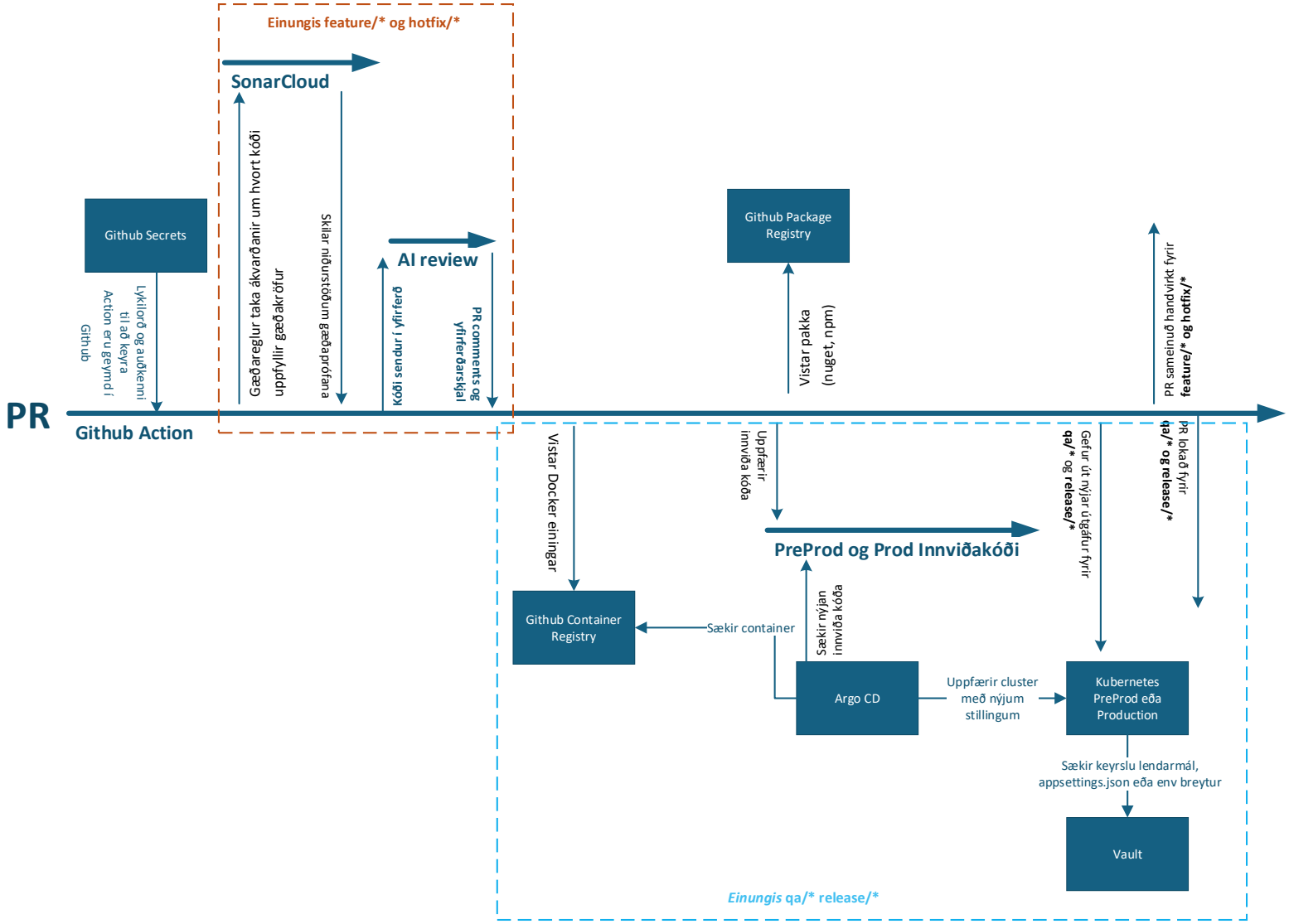
6.5 Reglur um innviðakóða og sjálfvirknivæðingu

Regla	**Lýsing**
Infrastructure as Code	Allar innviðaskilgreiningar skulu geymdar sem kóði — bæði Docker-skipanir og Kubernetes-skilgreiningar
Build í CI, ekki Docker	GitHub Actions píplínur sjá um build, compile og test. Docker einingar pakka einungis inn forsmíðuðum kóða
Einfaldur Docker	Dockerfile skal vera eins einföld og mögulegt er
Viðráðanlegur CI-kóði	Stór workflow skjöl skulu brotin upp í smærri, afmarkaðar einingar. Aðgerðir sem notaðar eru á mörgum stöðum skulu útfærðar sem endurnýtanlegar Actions

6.6 Útgáfuferill heildarmynd

Myndin sýnir heildarferli frá kóðaskilum til útgáfu í PreProd og Production, þar á meðal:

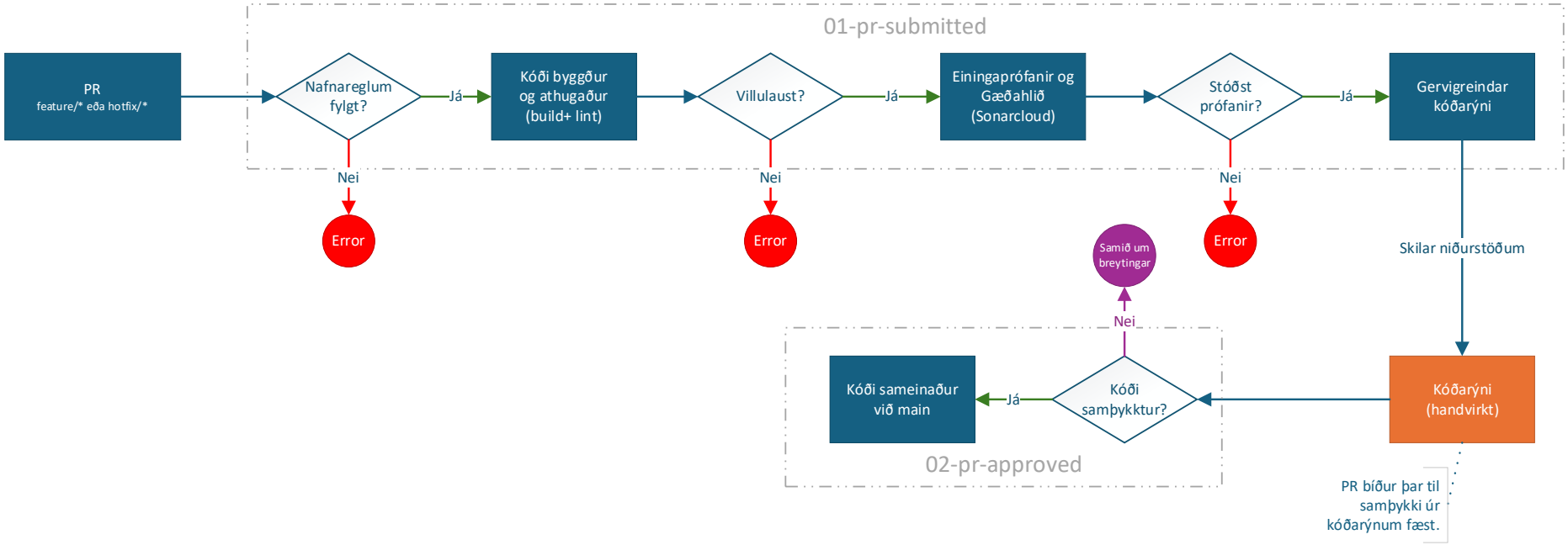
- Sjálfvirka gæðaferla við breytingabeiðnir (feature/* og hotfix/*)
- Docker-pökkun og geymsla í container registry verkkaupa
- Argo CD-samhæfingu og Kubernetes-útgáfu
- Leyndarmálasókn úr Vault



6.7 Útgáfuferill í skrefum

6.7.1 Feature og Hotfix kóðagreinar

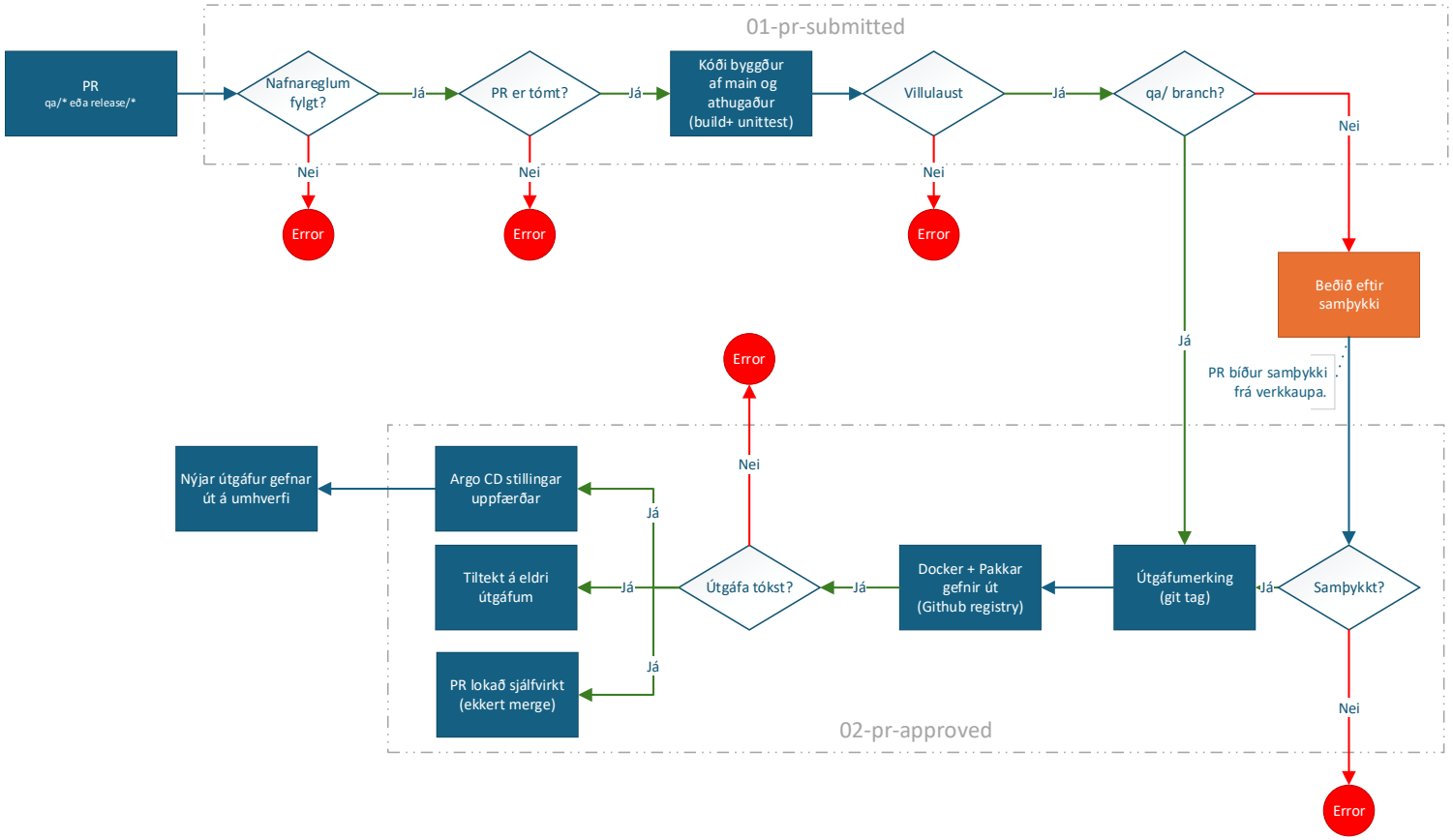
Ferilinn fyrir feature/* og hotfix/* breytingabeiðnir:



Skref	Lýsing	Afleiðing ef bregst
1	Nafnareglum fylgt?	PR hafnað
2	Kóði byggður og athugaður (build + lint)	PR hafnað
3	Einingaprófanir og gæðahlið keyrt	PR hafnað
4	Kóði sameinaður við main	—
5	Gervigreindarkóðarýni skilað niðurstöðum	—
6	Kóðarýni (handvirkt) og samþykkt	PR bíður samþykkis

6.7.2 QA og Release kóðagreinar

Ferilinn fyrir qa/* og release/* breytingabeiðnir:



Skref	Lýsing	Afleiðing ef bregst
1	Nafnareglum fylgt og PR er tómt?	PR hafnað
2	Kóði byggður af main og athugaður (build + unittest)	PR hafnað
3	Docker-einingar og pakkar gefnir út í container registry verkkaupa	PR hafnað
4	qa/ grein: sjálfvirk samþykkt. release/ grein: beðið eftir handvirkri samþykkt verkkaupa	PR bíður samþykkis (release)
5	Útgáfumerking (git tag)	PR hafnað ef merking mistekst
6	Argo CD-stillingar uppfærðar og ný útgáfa sett á umhverfi	—
7	PR lokað sjálfvirkt (ekkert merge). Tiltekt á eldri útgáfum	—

7. Viðauki – Útgáfunúmer og merking

Hugbúnaðurinn okkar skal notast við dagsetningarútgáfur (e. Calendar Versioning, http://Calver.org). Dagsetningarútgáfur gefa breiðari hópi notenda 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 Play 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.

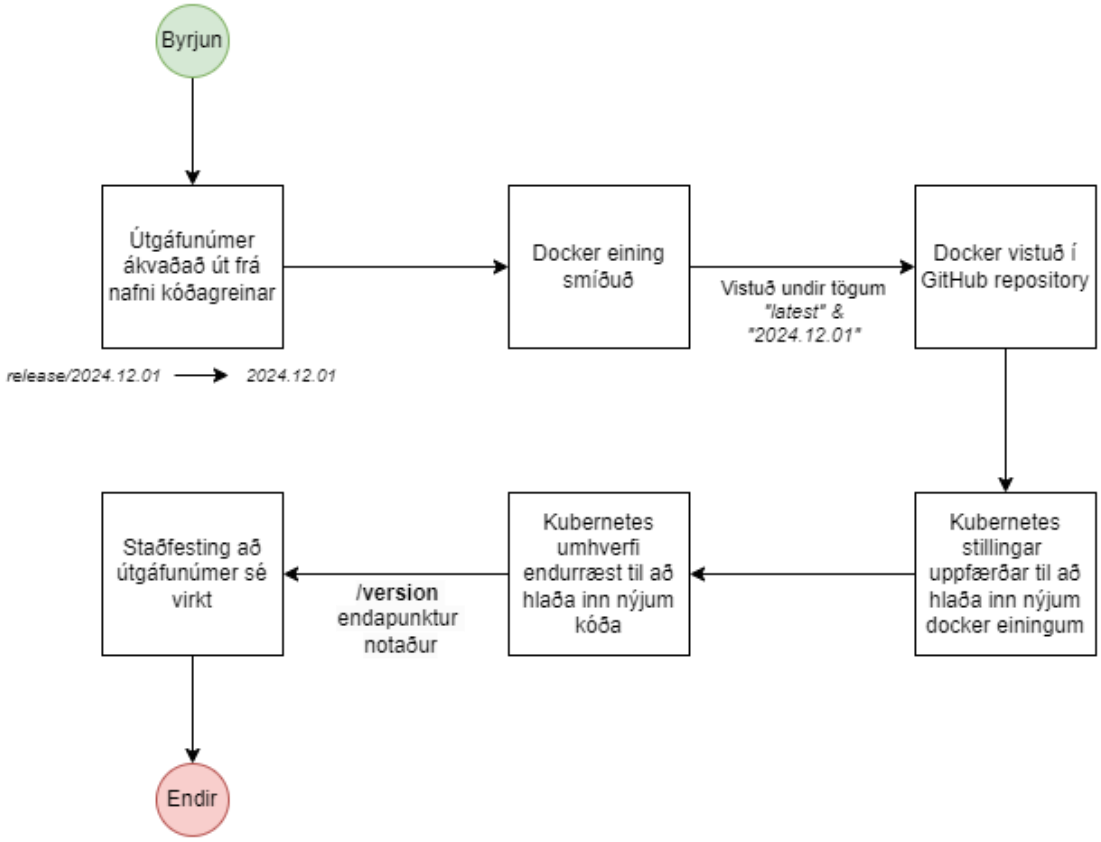
7.1 Útgáfumerking

Allur hugbúnaður sem ætlaður er til útgáfu á rekstrarumhverfum okkar er útbúinn til útgáfu með sjálfvirkum ferlum. Þetta á við um öll umhverfi, 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

7.2 Útgáfuferillinn í mynd



8. Viðauki – Viðmiðunararkitektúr, þjónustur og dæmi um lausnahögun

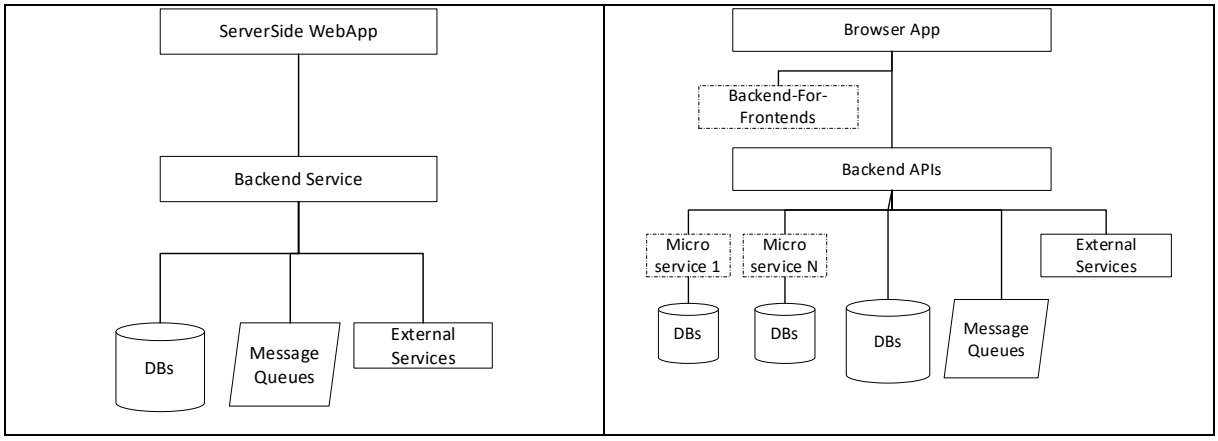
Kerfi smíðuð fyrir verkkaupa skulu styðja samvirkni við HL7 staðalinn í heilbrigðisgögnum og samþættingu við FHIR kerfiseiningar verkkaupa. Kerfi skulu reyna eftir fremsta megni að lesa, meðhöndla og miðla gögnum samkvæmt HL7 staðli eins og hann er skilgreindur á landsvísu og nota miðlægar FHIR innviðalausnir verkkaupa.

Kerfi skulu útfæra rótæklega 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 síðumiðaðar (e. page centric) lausnir sem krefjast ekki sérstakrar gagnvirkni[1] eða er ekki ætlað að hafa sérskilgreint langtíma viðhaldsteymi.

Æskilegt flækjustig í kerfishögun vefkerfa	Óæskilegt flækjustig í kerfishögun vefkerfa

[1] Dæmi um sérstaka gagnvirkni: drag-and-drop og sérhæfður innsláttur (töflureiknar, grafísk hönnun, textaritlar, spjallforrit); hugbúnaðarlausn sem þarf að virka ótengd (e. offline) eða í mjög slæmum tengiskilyrðum; gagnvirk kvikun/virkni (kortabirting, leikir, myndbönd, myndræn gagnaframsetning).



8.1 Ástæður fyrir rótækilega einfölduðum tækniarkitektúr

Verkkaupi aðhyllist rótæklega einfaldaðan tækniarkitektúr til að tryggja langtíma stöðugleika, fyrirsjáanleika og viðhaldshæfni hugbúnaðarlausna yfir líftíma sem spannar áratugi.

Rótæk einföldun felur í sér meðvitaða ákvörðun um að halda fjölda tæknieininga, forritunarlaga og samspils milli kerfishluta í lágmarki, án þess að skerða gæði, öryggi eða virkni lausna.

Með einföldum og samræmdum tæknistakk:

- styttist lærdómskúrfa nýs tæknifólks,
- dregur úr rekstraráhættu og bilanatíðni,
- einfaldast uppfærslur, öryggisviðbrögð og villuleit,
- minnkar háð við sérþekkingu einstaklinga eða tímabundnar tæknivinsældir.

Flóknar lausnir með mörgum sjálfstæðum einingum, umfangsmikilli client-side rökfræði eða tíðum tæknibreytingum leiða óhjákvæmilega til aukins viðhaldskostnaðar og rekstraráhættu til langs tíma. Slíkar lausnir samræmast ekki rekstrarforsendum verkkaupa.

Rótæk einföldun er því ekki val á „leiðinlegri tækni" af smekk, heldur stefnumótandi ákvörðun sem byggir á heildstæðu mati á rekstri, fjármögnun, mannauði og ábyrgð gagnvart framtíðarnotendum og rekstraraðilum.

8.2 Þjónustuhögun

Verkkaupi tekur ekki við hugbúnaði til reksturs sem samsettur er úr smáþjónustum (e. micro/nano services). Bakendakóði skal vera útfærður samkvæmt þjónustuhögun (e. service oriented architecture) en með áherslu á stærri þjónustueiningar.

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

Nánari upplýsingar um REPR högun:

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

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.

8.3 OpenAPI og forritunarskil

Allar þjónustur skulu hafa OpenAPI skilgreiningar sem lýsa forritunarskilum þeirra. Sjá viðauka „Kröfur til OpenAPI skilgreininga" fyrir fulla kröfulýsingu.

8.4 Gagnasnið

Gögn sem eru send og móttekin af þjónustum skulu vera á JSON sniði. Innri þjónustur geta nýtt sér gRPC/Protobuf í samskiptum sín á milli ef við á. Ekki skal gera greinamun á stórum og litlum stöfum í svæðaheitum í JSON skeytum (t.d. „data", „Data", „DATA" eru allt sama svæðið).

9. Viðauki - Þolgæði, dæmi og útfærslur

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.
- Útfærsla á samskiptum beint á móti HTTP klösum í Java og .NET (sbr HttpClient í .NET).

10. Viðauki - Skipulag prófana og tæknilegar útfærslur

Þessi viðauki lýsir tæknilegu skipulagi prófana, nafnareglum, möppuskipulagi prófunarkóða og reglum um undanskilinn kóða. Bindandi þekjumörk og gæðaviðmið eru skilgreind í kafla 13 (Hugbúnaðarprófanir) og viðauka „Kóðastíll og gæðatól."

10.1 Skipulag prófunarkóða

Einingaprófanir skulu vera geymdar í sér möppu undir Tests/, skipt upp í sér project (C#), sér rótarpökkum (Java) eða í sér rótarmöppum/namespace (önnur mál) eftir því hvers tegundar prófanirnar eru.

Prófunartegund	**Staðsetning**
Einingaprófanir	Tests/UnitTests/
Samþættingarprófanir	Tests/IntegrationTests/
Viðmótsprófanir	Tests/UITests/
Handvirkar prófanir	Tests/ManualTests/

Uppbygging einingaprófana skal endurspegla möppuskipulag lausnarinnar, en hafa rótarmöppuna Tests/UnitTests/.

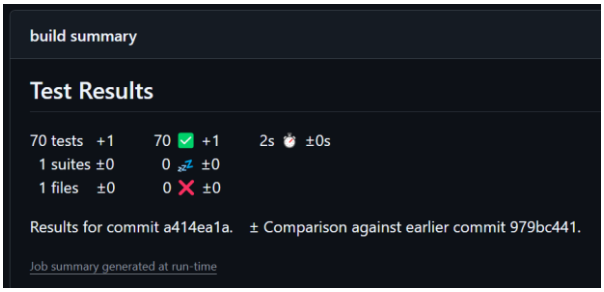
Uppbygging samþættingarprófana skal endurspegla ytra skipulag lausnarinnar (URL path), en hafa rótarmöppuna Tests/IntegrationTests/.

10.2 Prófunargögn

Prófunargögn fyrir einingaprófanir skulu fylgja forritunarkóða og vera hluti af einingaprófunarverkefninu. Prófunargögn skulu sitja undir rótarmöppu verkefnisins í Tests/UnitTests/TestData og endurspegla möppuskipulag lausnarinnar.

10.3 Niðurstöður prófana

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.



Mynd: Dæmi um Test Results samantekt úr GitHub Actions

10.4 Samþættingarprófanir

Öll forritunarskil (APIs) skulu innihalda viðeigandi samþættingarprófanir fyrir alla endapunkta sem eru aðgengilegir.

Þjónustur skulu vera prófanlegar á sjálfvirkan hátt óháð ytri auðkenningarþjónustum.

11. Prófunarrammar

Einingapróf skulu vera skrifuð í viðurkenndu prófunarframeworki, t.d. xUnit, jUnit eða sambærilegu. Óheimilt er að nota heimasmíðað eða óreynt framework til einingaprófana sem hluta af hugbúnaðarlausninni, né nota framework sem var smíðað af verksala og ætlað fyrir önnur verkefni.

11.1.1 Nafnareglur prófana

Einingapróf skulu fylgja nafnahefðum eins og þær eru skilgreindar af Microsoft:
https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices#naming-your-tests

Nöfn skulu vera samansett af þremur hlutum:

Hluti	Lýsing	Dæmi
Nafn falls	Nafnið á fallinu sem verið er að prófa	Add
Tilvik	Tilvikið sem verið er að prófa	SingleNumber
Niðurstaða	Væntanleg niðurstaða úr prófinu	ReturnsSameNumber

Samansett: `Add_SingleNumber_ReturnsSameNumber`

11.2 Undanskilinn kóði frá einingaprófunum

Heimilt er að undanskilja afmarkaðan kóða frá einingaprófunum, enda liggi fyrir rökstutt og samþykkt mat tæknilegra sérfræðinga verkkaupa.

11.2.1 Kröfur um rökstuðning

Þegar kóði er undanskilinn frá einingaprófunum verður rökstuðningur (justification) að fylgja í kóðanum sjálfum. Rökstuðningur skal vera á ensku og lýsa á stuttan og skýran hátt hvers vegna kóðinn er undanskilinn.

C# dæmi:

```
[ExcludeFromCodeCoverage(Justification = "Class inherited from external
system, code not modified to maintain parity with original implementation")]
public class ExternalLegacyAdapter
{
    // ...
}
```

Dæmi um ásættanlegan rökstuðning:

Rökstuðningur	Hvenær á við
"Auto-generated code from OpenAPI specification"	Sjálfvirkt gerður kóði úr OpenAPI skilgreiningu
"Thin wrapper over external library with no custom logic"	Hjúpun á ytri safni án eigin rökfræði
"Class inherited from external system, code not modified"	Kóði yfirtekinn frá ytra kerfi, óbreyttur
"DI registration module with no testable logic"	DependencyInjection skráningarkóði

Dæmi um óásættanlegan rökstuðning:

Rökstuðningur	Hvers vegna óásættanlegt
"Too complex to test"	Flækjustig er ástæða til að prófa, ekki til að sleppa
"Will add tests later"	Prófanir skulu fylgja kóða við skil
"Not important"	Kóði sem er hluti af lausn er alltaf mikilvægur
(Enginn rökstuðningur)	Rökstuðningur er skylda

11.2.2 Eftirlit

Sjálfvirkir gæðaferlar verkkaupa geta greint ExcludeFromCodeCoverage notkun og flaggað tilvikum sem skortir rökstuðning eða nota óásættanlegan rökstuðning. Slík tilvik eru meðhöndluð í kóðarýni.

12. Viðauki – Stöðluð forritunarstöfn verkkaupa

Forritunarsöfnin eru gefin út í lokuðu GitHub Package Registry. Hægt er að fá upplýsingar um aðgang að þessu safni frá sérfræðingum eða verkefnastjórum verkkaupa. Eftirfarandi listi inniheldur einungis helstu söfn verkaupa og er ekki tæmandi.

12.1 Gagnagrunnstengingar og ORM

Tilgangur	Nafn	Lýsing
Gagnagrunnstengingar	library-dapper-package-nuget	Hjálparföll fyrir Dapper.NET gagnagrunnstengingar, þ.m.t. DI skráningu, bulk insert aðgerðir, ORM vörpun og endurteknir (retry) aðgerðir með Polly.NET

12.2 API og endapunktar

Tilgangur	Nafn	Lýsing
FastEndpoint viðbætur (REPR)	library-fastendpoints-package-nuget	Staðlaðar útfærslur fyrir FastEndpoints ramma, þ.m.t. DI skráningu, villuleit (validators), Swagger stillingar og leiðahjálpar
Version endapunktur (REPR)	library-fastendpoints-versionendpoint-package-nuget	Staðlaður /version endapunktur fyrir FastEndpoints verkefni sem birtir útgáfu og smíðaupplýsingar
Version endapunktur (RCL)	library-rcl-versionendpoint-package-nuget	Staðlaður /version endapunktur fyrir ASP.NET Razor verkefni
Version endapunktur (WebAPI)	library-versionendpoint-package-nuget	Staðlaður /api/version endapunktur fyrir Controller og Minimal API verkefni

12.3 Logging, mælingar og keyrslustaða

Tilgangur	Nafn	Lýsing
Structured Logging	library-serilog-package-nuget	Staðlaðar Serilog og Coralogix útfærslur með DI skráningu og sérsniðnum Coralogix sink
Afkastamælingar	library-prometheus-package-nuget	Prometheus mælingatöku fyrir gagnagrunna, skyndiminni og API beiðnir með stöðluðum endapunktum
Stöðuendapunktar (K8s)	library-healthchecks-package-nuget	Staðlaðar ASP.NET heilsuathuganir með /system/ready og /system/alive endapunktum

12.4 ASP.NET middleware og pipeline

Tilgangur	Nafn	Lýsing
ASP Request Pipeline þjónustur	library-aspmiddleware-package-nuget	ASP.NET middleware fyrir fylgni-ID, aðgerða-ID, lotu-ID og X-Road hausum, auk villumeðhöndlunar

12.5 Prófanir

Tilgangur	Nafn	Lýsing
Einingaprófanir	library-unittestframework-package-nuget	Hjálparföll fyrir einingaprófanir með Moq útfærslum fyrir Dapper og aðgang að csproj stillingum

12.6 Auðkenning og heimildir

Tilgangur	Nafn	Lýsing
API-lykla auðkenning	library-apikeyauthentication-package-nuget	Staðlað API lykla auðkenningarkerfi fyrir ASP.NET Core með skyndiminni og gagnagrunnstengingu
Rafræn skilríki	library-islandisauthentication-package-nuget	Island.is OpenID auðkenningarflæði fyrir Razor og MVC verkefni með kökustjórnun
Grunnauðkenning	library-coreauthentication-package-nuget	Fjölstefnu auðkenning með kvikri stefnuvali og staðlaðri kökuauðkenningu
Saga auðkenning	library-sagaauthentication-package-nuget	Auðkenning gegnum Saga JWT tóka og Hekla lotukökur fyrir samþættingu við sjúkraskrákerfi
Heimildaþjónusta SDK	library-authorizationservicesdk-package-nuget	SDK til að tengjast heimildaþjónustu API með DI skráningu og stillingum

12.7 DI og arkitektúr

Tilgangur	Nafn	Lýsing
Dependency Injection og ServiceRepository högun	library-dependencyinjection-package-nuget og library-dependencyinjection-abstractions-package-nuget	Stöðluð útfærsla á ServiceRepository högun. Allar þjónustur skulu nýta sér þessa högun til að forðast miðstýringu á DI skilgreiningum.
Stöðluð grunnvirkni	library-standardproviders-package-nuget	DI-inndælanlegir veitendur fyrir DateTime, Guid, Stopwatch, JSON og öryggisaðgerðir

12.8 Notendaviðmót og framendi

Tilgangur	Nafn	Lýsing
Notendaviðmót — almennir íhlutir	library-landlaeknirui-package-nuget	TagHelper útfærslur fyrir UI íhlutasafn verkkaupa ætlað almennum lausnum (t.d. takkar, hleðsluvísar, inntaksreitir). *Athugið: nafn pakka mun breytast.*
Notendaviðmót — íhlutir fyrir heilbrigðisstarfsfólk	library-landlaeknirui-hcp-package-nuget	TagHelper útfærslur fyrir UI íhlutasafn verkkaupa ætlað lausnum fyrir heilbrigðisstarfsfólk (HCP). *Athugið: nafn pakka mun breytast.*
HTMX og Razor samvirkni (RCL)	library-asprclextensions-package-nuget	HTMX og Razor hjálparföll, þ.m.t. ViewBag viðbætur, QueryString aðstoð o.þ.h.
Kóðaaðstoð (RCL)	library-asprclsourcegenerators-package-nuget	Sjálfvirkir kóðaframleiðendur fyrir Razor Class Libraries sem búa til sterktegunduð nöfn

Þýðingar (RCL)	library-asprcllocalization-package-nuget	Stuðningur við staðfærslu (localization) í Razor Class Libraries með sérsniðnum resource localizer

12.9 Þolgæði og HTTP

Tilgangur	Nafn	Lýsing
Þolgæðisútfærslur og HTTP viðbætur	library-httpextensions-package-nuget	HTTP viðbætur með Polly.NET endurtekningu, ProblemDetails stuðningi og netþjónastillingum

12.10 Skipanalínuviðmót

Tilgangur	Nafn	Lýsing
Skipanalínuviðmót	library-spectre-package-nuget	Spectre.Console útfærslur fyrir CLI forrit með litaðri útprentun og framvinduvísum

13. Viðauki - Geymsla leyndarmála

Þessi viðauki lýsir hvernig skuli meðhöndla leyndarmál (secrets) í þróun, sjálfvirknivæðingu og keyrslu hugbúnaðarlausna fyrir verkkaupa. Markmiðið er að tryggja öryggi, einfaldleika og fyrirsjáanleika, án þess að búa til óþarfa flækjustig í þróun eða rekstri.

13.1 Hvar eru leyndarmál geymd?

Samhengi	Lausn
CI ferlar (GitHub Actions)	GitHub Secrets
Keyrsla í Kubernetes	HashiCorp Vault
Forritunarkóði	Aldrei
Docker images	Aldrei

13.2 GitHub Actions (CI ferlar)

Í sjálfvirknipípum í GitHub skal notast við GitHub Secrets.

- Fyrst skal meta hvort hægt sé að geyma secrets á organization-stigi, þannig að:
    - fleiri en ein kóðageymsla geti samnýtt gildin,
    - samræmi náist milli verkefna.
- Ef það er ekki mögulegt, má nota secrets á repository-stigi.

GitHub Secrets eru eingöngu ætluð í build ferlum, prófunum og pakkasmíði.

Þau skulu aldrei:

- vera skrifuð í skrár,
- send áfram í Docker images,
- notuð beint í keyrsluumhverfi Kubernetes.

13.3 Kubernetes og Vault (keyrsluumhverfi)

Allar lausnir sem keyra í Kubernetes skulu nota HashiCorp Vault, sem er uppsett fyrir hvert Kubernetes cluster.

Engin leyndarmál skulu vera:

- í Kubernetes YAML skrám,
- í GitHub,
- í Docker images,
- í environment breytum sem eru committed.

Leyndarmál eru lesin úr Vault rétt áður en þjónusta er ræst.

Vault er eini sannleikurinn (single source of truth) fyrir:

- lykilorð,
- API-lykla,
- tengistrengi,
- vottorð og önnur viðkvæm gildi.

13.4 .NET lausnir – appsettings.json (aðalleið)

Fyrir allar nýjar lausnir (og endurbætur á eldri) er eftirfarandi leið sjálfgefin og æskileg.

- Allar stillingar, þar með talin leyndarmál, eru skilgreindar í appsettings.json.
- Engar sértækar environment breytur eru notaðar til að sækja stillingar, nema sérstök krafa sé um annað.
- Leyndarmál eru ekki brotin niður í stök gildi, heldur geymd sem heilt appsettings.json skjal.

13.4.1 Geymsla í Vault

Fyrir hvert umhverfi (PreProd, Production) er geymt sérstakt appsettings.json í Vault.

Vault-slóð skal vera á eftirfarandi formi:

services/data/NAFN-Á-ÞJÓNUSTU/appsettings.json

13.4.2 Hleðsla í forriti

.NET þjónustur skulu reyna að hlaða inn yfirskriftarskrá (appsettings.override.json) ef hún er til staðar.

Dæmi (Program.cs):

```
builder.Configuration
       .AddJsonFile("appsettings.json", optional: false)
       .AddJsonFile("appsettings.override.json", optional: true)
       .AddEnvironmentVariables();
```

Athugið: Environment variables eru ekki hluti af sjálfgefnu mynstri nema verkefni krefjist þess sérstaklega.

13.4.3 Kubernetes keyrsla

Í Kubernetes stillingum er ENTRYPOINT þjónustunnar yfirskrifað þannig að appsettings.json úr Vault sé skrifað sem appsettings.override.json inn í container áður en þjónustan er ræst.

Dæmi:

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

13.4.4 Environment breytur – takmörkuð notkun

Environment breytur skulu aðeins notaðar:

- þegar um er að ræða almenn, óviðkvæm gildi, eða
- þegar ytri tækni eða rekstrarkröfur gera það óhjákvæmilegt.

Ef environment breytur eru notaðar:

- Sama nafn skal notað í öllum umhverfum (t.d. API_URL, ekki API_URL_PROD, API_URL_TEST).
- Óviðkvæm gildi skulu vera skilgreind í Kubernetes stillingum (values.yaml o.þ.h.).
- Environment breytur skulu aldrei innihalda:
  - lykilorð,
  - API-lykla,
  - tengistrengi,
  - önnur viðkvæm gögn.

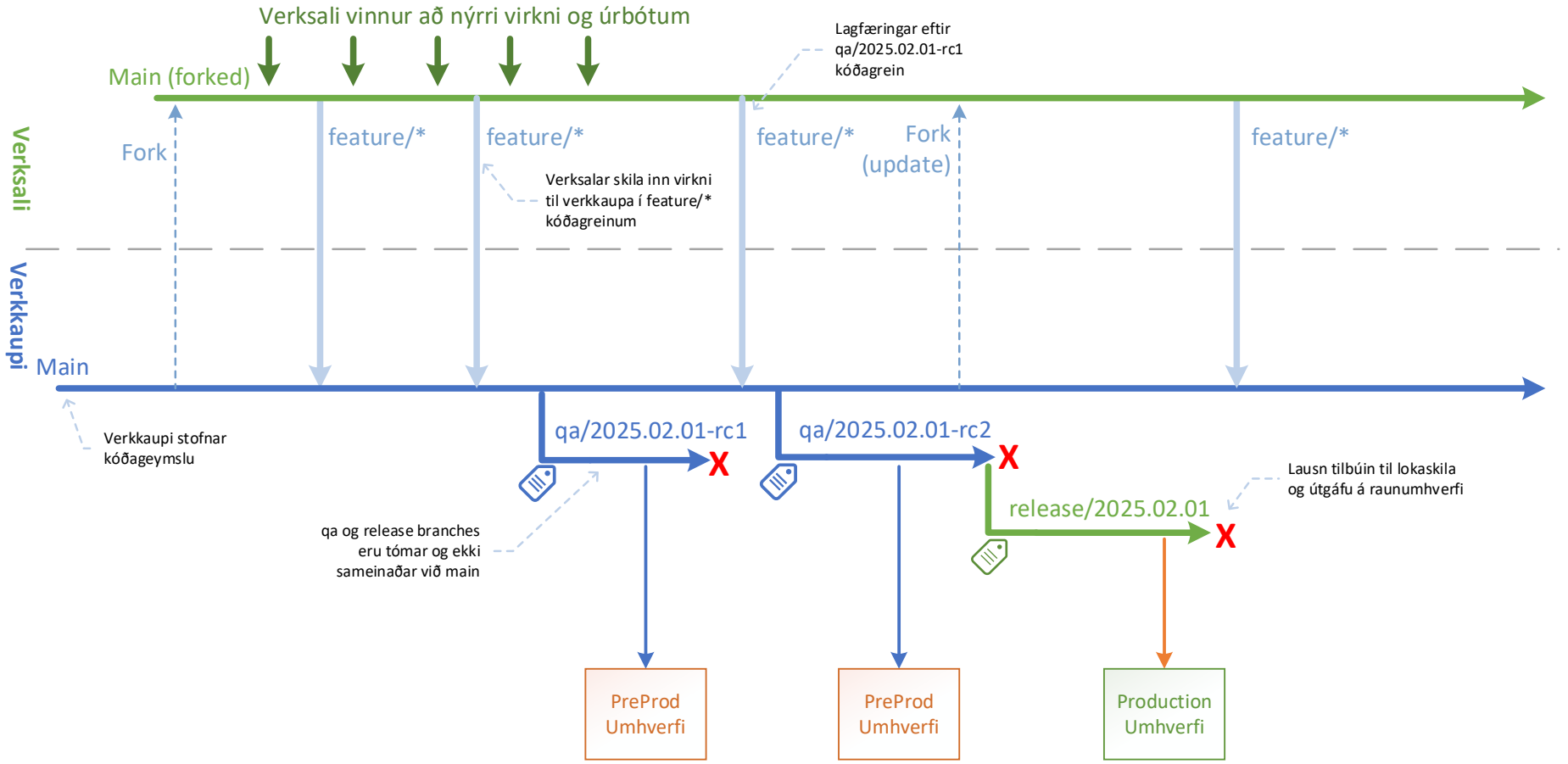
13.5 Legacy lausnir (fyrir tíma þróunarhandbókar)

Þessi kafli á eingöngu við eldri lausnir sem voru smíðaðar áður en núverandi þróunarhandbók tók gildi.

- Eldri JavaScript eða önnur legacy kerfi mega áfram nota environment breytur samkvæmt eldri mynstrum.
- Slíkar lausnir skulu:
    - sækja allar stillingar á einum stað í kóða,
    - safna þeim í immutable configuration object,
    - ekki lesa environment breytur dreift um kóðann.

Allar nýjar lausnir skulu fylgja .NET / appsettings / Vault leiðinni.

14. Viðauki - Samspil kóðagreina og skila



Útgáfa á Keyrslu umhverfi er byggt á nafni á kóðagrein

Í þeim tilvikum þar sem þörf er á að halda utan um margar langlífar útgáfugreinar (t.a.m. þar sem samþætt er við ytri hugbúnaðarútgáfur) þá skal viðhalda einni aðal grein fyrir hverja major útgáfu en annars fylgja sama ferli. Nýjasta major útgáfa skal ávallt vera á main grein. *Sjá EHDS verkefni.*

15. Viðauki – Kóðastíll og gæðatól – verkfæri, stillingar og reglusett

Þessi viðauki lýsir þeim verkfærum, stillingum og reglusettum sem verkkaupi notar til að tryggja samræmi og gæði í forritunarkóða. Viðaukinn er ætlaður forriturum og tæknifólki verksala til hliðsjónar við uppsetningu þróunarumhverfa.

15.1 Sjálfvirk samræming og kóðastíll (linting)

Allur kóði sem skilað er til verkkaupa fer í gegnum sjálfvirka samræmingu og stílaðlögunarforrit (linting). Verkkaupa notar eftirfarandi tól:

Tól	Hlutverk	Tengill
SonarCloud	Miðlæg kóðagreining, gæðahlið og öryggisgreining	https://www.sonarsource.com/products/sonarcloud/
SonarLint	Staðbundin kóðagreining í þróunarumhverfi forritara	https://www.sonarsource.com/products/sonarlint/ide-login/

Mælt er með að allir forritarar verksala tengi SonarLint við SonarCloud verkefni verkkaupa til að fá samræmda endurgjöf á meðan þróun stendur.

15.2 C# (aðalforritunarmál)

Tól / Reglusett	Lýsing
Roslyn Analyzers (StyleCop)	Kóðastílsgreining og arkitektúrreglur
global.ruleset	Grunnstillingar frá verkkaupa; notaðar sem upphafspunktur

Verksali fær aðgang að global.ruleset stillingum verkkaupa við upphaf verkefnis.

Heimilt er að slökkva á ákveðnum stílreglum ef þörf krefur, en slíkt skal gerast í samráði við sérfræðinga verkkaupa og vera skjalfest.

15.3 TypeScript / JavaScript

Eftirfarandi kröfur gilda um TypeScript og JavaScript kóða í þremur tilvikum: (a) eldri lausnir sem voru smíðaðar fyrir tíma þessarar þróunarhandbókar, (b) lausnir sem hafa fengið formlega undanþágu, og (c) Leið B framendaverkefni samkvæmt kafla 8.2.2.

15.3.1 ESLint – grunnreglusett

Tól / Reglusett	Lýsing	Á við
ESLint	Kóðastílsgreining	Allar TS/JS lausnir
eslint-config-airbnb	Grunnreglusett	Allar TS/JS lausnir
eslint-config-airbnb-typescript	Viðbót fyrir TypeScript	Allar TS/JS lausnir

15.3.2 ESLint – viðbótarreglur fyrir Leið B (React) verkefni

Leið B verkefni (kafli 5.2.2) skulu einnig nota eftirfarandi ESLint viðbætur:

Tól / Reglusett	Lýsing
eslint-plugin-react	React-sértækar reglur
eslint-plugin-react-hooks	Reglur fyrir rétta notkun React Hooks
eslint-plugin-jsx-a11y	Aðgengisreglur fyrir JSX

Allar React-sértækar ESLint reglur skulu vera virkjar sem villur (errors), ekki viðvaranir. Undantekningar skulu vera skjalfestar og samþykktar af verkkaupa.

15.4 Meginregla um viðvaranir

Öll samræmingar- og stílaðlögunarforrit skulu vera stillt þannig að viðvaranir séu meðhöndlaðar sem villur (*treat warnings as errors*).

Þetta á við um:

- þýðanda (compiler),
- kóðagreiningu (static analysis),

- stílreglur (linting).

Mælt er með að forritarar verksala stilli þróunarumhverfi sín á sama hátt til að forðast óþarfa endurvinnu við skil.

15.5 Þýðistillingar (compiler settings)

Eftirfarandi stillingar eru prófaðar sjálfvirkt af gæðaferlum verkkaupa við hverja breytingabeiðni. Verksali ber ábyrgð á að þessar stillingar séu til staðar og réttar í lausn sinni.

15.5.1 C# – .csproj eða Directory.Build.props

Stilling	Krafa
TreatWarningsAsErrors	true
CodeAnalysisRuleSet	Til staðar; vísar í global.ruleset frá verkkaupa
GenerateDocumentationFile	Til staðar

Eftirfarandi NuGet pakkar skulu vera hluti af lausn:

Pakki	Tilgangur
StyleCop.Analyzers	Stílgreining
StyleCop.CSharp.Async.Rules	Async-sérstakar stílreglur
Microsoft.CodeAnalysis.NetAnalyzers	.NET greiningarreglur

15.5.2 TypeScript – tsconfig.json

Eftirfarandi stillingar gilda um allar lausnir sem innihalda TypeScript kóða, þar á meðal eldri lausnir, lausnir með formlega undanþágu og Leið B framendaverkefni (kafli 8.2.2).

Eftirfarandi stillingar skulu allar vera settar á true:

Stilling	
strict	noImplicitReturns
alwaysStrict	noUnusedLocals
noImplicitAny	noUnusedParameters
strictNullChecks	useUnknownInCatchVariables
strictFunctionTypes	sourceMap
exactOptionalPropertyTypes	noUncheckedSideEffectImports

15.5.3 Viðbótarreglur fyrir Leið B (React) verkefni:

Leið B verkefni skulu einnig setja eftirfarandi stillingar í tsconfig.json:

Stilling	Gildi	Ástæða
jsx	react-jsx	Nauðsynlegt fyrir React JSX meðhöndlun
module	ESNext	Vite notar ESModule
moduleResolution	bundler	Samhæfni við Vite smíðaverkfæri
isolatedModules	true	Tryggir samhæfni við Vite/esbuild þýðingu
target	ES2020 eða nýrra	Nútíma vafrastuðningur

Viðbótarreglur (óbreyttar frá fyrri útgáfu):

- Ekki má slökkva á undirstillingum sem heyra undir yfirstillingar (t.d. ekki setja strict=true og svo strictFunctionTypes=false).
- Ef allowJS er sett á true verður checkJs einnig að vera true.

15.6 Gæðahlið – sjálfgefin viðmið

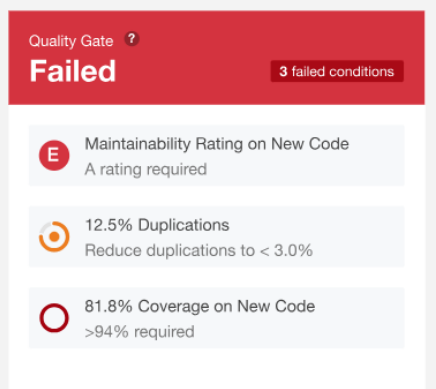
Eftirfarandi gæðaviðmið eru sjálfgefin í gæðahliðum verkkaupa og gilda fyrir allar lausnir nema annað sé sérstaklega samþykkt:

Viðmið	Gildi
Einingaprófanir — þekja á nýjum kóða	≥ 80%
Einingaprófanir — heildarþekja	≥ 65%; má ekki lækka milli útgáfa
Viðvaranir í kóðagreiningu	Meðhöndlaðar sem villur
Alvarlegar Docker öryggisathugasemdir	Engar á „Critical" stigi

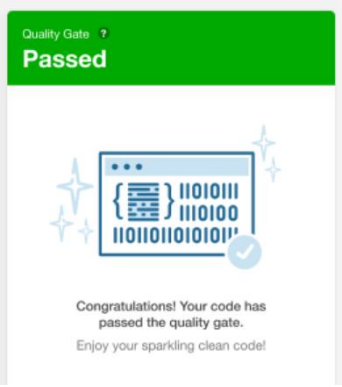
Gæðahlið nota *clean-as-you-code* aðferðarfræði: kóði í breytingabeiðni er borinn saman við núverandi stöðu í kóðagrunninum. Þetta tryggir að gæði batni jafnt og þétt án þess að leggja á verksala vandamál í ótengdum eldri kóðahlutum.

15.6.1 Dæmi um gæðahliðsniðurstöður

Eftirfarandi skjámyndir sýna dæmi um niðurstöður gæðahliðs í kóðagreiningu verkkaupa.



Gæðahlið sem hefur hafnað kóðaskilum



Gæðahlið sem samþykkti kóðaskil

Ef gæðahlið hafnar breytingabeiðni uppfyllir kóðinn ekki lágmarks gæðakröfur og skilum er hafnað. Verksali ber ábyrgð á að lagfæra og endursenda.

15.7 Öryggisprófanir á Docker einingum

Allar Docker einingar fara í gegnum sjálfvirkar öryggisprófanir. Breytingabeiðni er hafnað ef athugasemdir finnast á hæsta viðvörunarstigi (Critical).

Tegund prófunar	Lýsing
Grunneiningar (base image)	Greining á grunn Docker einingum sem notaðar eru
Þriðja aðila pakkar	Greining á stýrikerfispökkum og forritunarsöfnum sem grunneiningar stóla á

Viðbótarprófanir geta verið keyrðar í ákveðnum kringumstæðum að mati verkkaupa.

16. Viðauki – Lagskipting og möppuskipan lausna

Þessi viðauki skilgreinir staðlaða rótarmöppuskipan og lagskiptingu forritunarkóða í hugbúnaðarlausnum verkkaupa. Allur forritunarkóði skal vera skipulagður undir þessum rótarmöppum. Enginn kóði skal liggja beint undir rót kóðageymslu nema þar sem sérstaklega er þörf og skilgreint (t.d. src/).

16.1 Rótarmöppuskipan

Mappa / Skrá	Á við um	Lýsing
Clients/	Allt	Entry points hugbúnaðarins (exe, entry dll). Inniheldur þau project sem útfæra aðalræsingu.
Core/	Allt	Kjarnavirkni og innsta lag domain lagskiptingarinnar. Contracts (request/response), sameiginleg configuration og interfaces. Hvatt er til að aðgreina contract klasa í sér möppu eða project.
DependencyInjection/	Allt	Kóði sem tengist Dependency Injection og IoC. Skráning á einingum og týpum.
Docs/	Allt	Skjölun fyrir lausnina og allar myndir sem henni tengjast.
Endpoints/ eða Controllers/	Bakendi	Ysta lag beiðna og svara (request/response). Ef REPR högun er notuð skal nota Endpoints/; ef MVC högun er notuð skal nota Controllers/. Skrár skulu flokkaðar í möppur eftir virknisviði.
Pages/	Framendi (SSR)	Eiginlegar síður eða viðmót í framendalausninni. Aðgreindar í ASP.NET Areas fyrirkomulag og Razor Class Library (RCL) högun.
Infrastructure/	Allt	Hjúpun á ytri forritunarsöfnum. Project skulu byrja á forskeytinu Library. og hjúpa þriðja aðila forritunarsöfn.

Managers/	Allt	Flæðistjórnunarkóði — límið milli Endpoints/ og Repositories/. Inniheldur klasa sem lýsa aðgerðarflæði, nota Repository til að sækja gögn og umbreyta í domain módel.
Repositories/	Allt	Gagnasókn í ytri kerfi. Project flokkuð eftir gagnaveitu: DataAccess (gagnagrunnar), ApiAccess (ytri API), StreamAccess / QueueAccess (biðraðir), FileAccess (skrár). Vinnur með data módel en ekki domain módel.
Services/	Allt	Þjónustuklasar: bakgrunnsþjónustur, möppunarþjónustur og álíka sértæk virkni sem notuð er af Manager lagi.
SQL/	Bakendi	SQL kóði sem fylgir lausninni. Skipt upp eftir tilgangi og númerað eftir keyrslurröð (sjá viðauka um gagnagrunnskóða).
Tests/	Allt	Prófanir: Tests/UnitTests/, Tests/IntegrationTests/, Tests/UITests/ og aðrar tegundir. Uppbygging prófana skal endurspegla möppuskipulag lausnarinnar.
.github/	Allt	CI/CD sjálfvirkni (GitHub Actions workflows).
.kube/	Allt	Argo CD innviðakóði. Möppuuppbygging skal endurspegla GitOps repo skipulag.
.gitignore	Allt	Hvaða skrár eiga ekki að fara í git.
.dockerignore og Dockerfile	Allt	Docker pökkunarskipanir.
SonarLint.xml og global.ruleset	Allt	Linting og stillingar á gæðaeftirliti kóða.
README.md	Allt	Lýsing á hvað kóðageymslan hefur að geyma (sjá kafla *Hugbúnaðarskjölun*).

16.2 Meginreglur lagskiptingar

Hvert lag skal hafa afmarkað hlutverk og vera eins sjálfstætt og mögulegt er. Forðast skal hringháð (circular dependencies).

Eftirfarandi lagskipting gildir:

Lag	**Hlutverk**	**Má kalla á**
Clients/	Ræsing og uppsetning	Öll lög
Endpoints/ eða Controllers/	Móttaka beiðna, validation, HTTP samningar	Managers/
Managers/	Flæðistjórnun, viðskiptareglur	Repositories/, Services/
Services/	Hjálparþjónustur, möppun, bakgrunnsvinnslur	Repositories/, Core/
Repositories/	Gagnasókn og gagnaskrif	Infrastructure/, Core/
Infrastructure/	Hjúpun á ytri söfnum	Core/
Core/	Contracts, interfaces, skilgreiningar	Ekkert (innsta lag)

Viðskiptaflæði og kjarna-rökfræði (Managers/, Core/) skulu aldrei vera háð ysta lagi (Endpoints/, Clients/) eða gagnalagi (Repositories/).

16.3 Óheimil hugtök í skipulagi - Leið A

Eftirfarandi hugtök og uppbygging sem tengjast SPA- eða client-miðaðri arkitektúr skulu ekki koma fyrir í Leið A lausnum:

Óheimilt	**Ástæða**
Hooks/ sem arkitektúrhugtak	React/SPA hugtak — á ekki við Leið A
Client-side state-lög	Samræmist ekki SSR stefnu
Framendaskipan sem byggir á Angular, Vue eða Svelte	Óheimilar tæknilausnir

16.4 Möppuskipan – Leið B

Leið B framendaverkefni (kafli 8.2.2) skulu fylgja einföldu og fyrirsjáanlegu möppuskipulagi sem endurspeglar React-staðla. Eftirfarandi skipulag er ráðlagt:

Mappa	**Innihald**
src/components/	React-íhlutir (components), skipulagðir eftir virknisviði
src/pages/ eða src/views/	Síður eða yfirlitslög
src/hooks/	Sérsniðin React hooks (heimilt í Leið B)

src/services/ eða src/api/	OpenAPI-mynduð eða handskrifuð API-þjónustulög
src/locales/	Staðfæringaskrár (i18n)
src/types/	Sameiginlegar TypeScript-týpur
public/	Statískar skrár (favicon, myndir o.fl.)
Tests/	Prófanir (Vitest/Jest)

Skipulagið skal vera einfalt og flatt. Djúpt möppuskipulag (meira en 3 stig) er óæskilegt og bendir til flækjustigs sem samræmist ekki kröfum um flækju við endursmíði í Leið B.

Eftirfarandi hugtök og uppbygging eru óheimil í Leið B lausnum:

Óheimilt	**Ástæða**
Store/, Redux/, Zustand/ eða álíka stöðustjórnunarlög	Ytri stöðustjórnunarsöfn óheimil í Leið B
Server/, API/ eða middleware/ lög sem vísa til Node.js bakenda	Enginn Node.js bakendi í Leið B
pages/api/ eða app/api/ (Next.js API routes skipulag)	Next.js óheimilt

16.5 Skipulag prófunarkóða

Prófunarkóði skal vera skipulagður undir Tests/ og endurspegla uppbyggingu lausnarinnar:

Prófunartegund	**Staðsetning**	**Uppbygging**
Einingaprófanir	Tests/UnitTests/	Endurspeglar möppuskipulag lausnar
Samþættingarprófanir	Tests/IntegrationTests/	Endurspeglar ytra skipulag (URL paths)
Viðmótsprófanir	Tests/UITests/	Skipulagt eftir viðmótssvæðum
Handvirkar prófanir	Tests/ManualTests/	Afrit af handvirkum prófunarskjölum ef við á
Prófunargögn	Tests/UnitTests/TestData/	Endurspeglar möppuskipulag lausnar

17. Viðauki – Kröfur til OpenAPI skilgreininga

Þessi viðauki lýsir kröfum til OpenAPI skilgreininga sem allar bakendaþjónustur verkkaupa skulu uppfylla. Skilgreiningarnar eru grundvöllur samþættingar, sjálfvirkrar prófunar, kóðagerðar og skjölunar.

17.1 Lágmarkskröfur

Allar OpenAPI skilgreiningar skulu að lágmarki innihalda eftirfarandi:

Þáttur	**Krafa**	**Lýsing**
Endapunktar	Allir skjalaðir	Allir aðgengilegir endapunktar skulu vera skilgreindir í OpenAPI skjalinu
Inntaksgögn	Lýst með JSON Schema	Allar beiðnir skulu hafa skilgreindan JSON Schema eða álíka formúlu
Skilagögn	Lýst með JSON Schema	Öll svör skulu hafa skilgreindan JSON Schema fyrir hverja HTTP stöðu
Gildissvið og takmarkanir	Skilgreindar	Lágmarks-/hámarksgildi, lengd strengja, skyldureitir og önnur viðmið
Villukóðar	Allir skjalaðir	Allir HTTP villukóðar sem þjónustan getur skilað, ásamt merkingu og sniði villusvars
Dæmi	Fyrir beiðnir og svör	Að lágmarki eitt dæmi per endapunkt sem sýnir dæmigerðan kall og svar
Auðkenningarmáti	Skjalfestur	Hvaða auðkenningarmáti er notaður (JWT, API-lykill o.þ.h.)
Aðgangsstýring per endapunkt	Skjalfest	Hvort endapunktur er aðgangsstýrður og hvaða heimildir (claims) eru nauðsynlegar

17.2 Gæðakröfur

OpenAPI skilgreiningin skal vera:

Gæðakrafa	**Lýsing**

Nákvæm	Nægilega nákvæm til að styðja sjálfvirka kóðagerð (code generation) og prófanir
Uppfærð	Endurspegla raunverulega virkni þjónustunnar á hverjum tíma
Samræmd	Fylgja samræmdum nafnareglum og sniðum innan lausnar og milli lausna verkkaupa
Prófanleg	Þjónusta skal vera prófanleg á sjálfvirkan hátt út frá OpenAPI skilgreiningu án háðs við ytri auðkenningarþjónustur

17.3 Gagnasnið

Krafa	Lýsing
JSON	Gögn sem send og móttekin eru skulu vera á JSON sniði
gRPC/Protobuf	Heimilt milli innri þjónusta ef við á
Case-insensitivity	Ekki skal gera greinamun á stórum og litlum stöfum í svæðaheitum JSON skeyta (t.d. „data", „Data", „DATA" eru allt sama svæðið)

17.4 Notkun í sjálfvirkni

OpenAPI skilgreiningar skulu nýttar til:

- sjálfvirkrar prófunar á samþættingu milli þjónusta,
- kóðagerðar (code generation) á client-söfnum,
- skjölunar sem birtist forriturum og samþættingaraðilum,
- sannreyningu á samræmi milli skilgreiningar og raunverulegrar útfærslu.

Verkkaupi getur gert kröfu um að OpenAPI skilgreining sé hluti af sjálfvirkum gæðaferlum og að frávik milli skilgreiningar og útfærslu leiði til höfnunar breytingabeiðni.

18. Viðauki – Skipulag og snið hugbúnaðarskjölunar

Þessi viðauki lýsir skipulagi, sniði og innihaldsviðmiðum hugbúnaðarskjölunar sem fylgir lausnum verkkaupa. Skjölunin er metin sjálfvirkt og handvirkt við hverja afhendingu (sjá kafla *Gæðaferlar* og viðauka *Yfirferðarskjal kóðaryni*).

18.1 Möppuskipulag skjölunar

Öll skjölun sem ekki er README.md skal geymd í sérstakri /Docs möppu í rót kóðageymslu.

```
/
├── README.md                  # Aðalinngangsskjal lausnar
├── Docs/
│   ├── INDEX.md               # Yfirlitsskjal sem vísar í öll undirskjöl
│   ├── img/                   # Allar myndir sem notaðar eru í skjölun
│   │   ├── architecture.png
│   │   └── ...
│   ├── architecture.md        # Arkitektúrlýsing (ef stærri en README rúmar)
│   ├── deployment.md          # Útgáfu- og keyrsluumhverfislýsing
│   ├── database.md            # Gagnagrunnslýsing (ef við á)
│   ├── integrations.md        # Ytri tengingar og samþættingar (ef við á)
│   ├── cronjobs.md            # Bakgrunnsvinnslur og tímasett verk (ef við á)
│   └── ...
```

18.2 Kröfur til /Docs möppu

Krafa	Lýsing
INDEX.md	Skylda. Yfirlitsskjal sem lýsir efni /Docs möppu og vísar með tenglum í öll undirskjöl.
img/	Skylda ef myndir eru notaðar. Allar myndir á PNG eða JPG sniði.
Efnisskrá (TOC)	Hvert Markdown skjal skal innihalda efnisskrá nálægt byrjun skjalsins með klikanlegum tenglum á helstu hluta.
Einungis 2 stig í TOC	Efnisskrá skal einungis innihalda fyrstu tvö stig fyrirsagna.

18.3 README.md – rótarskjal

README.md í rót kóðageymslu er aðalinngangsstaður lausnar. Það skal innihalda eftirfarandi, í þessari röð:

Hluti	Lýsing
Heiti lausnar	Nafn kóðageymslu eða lausnar
Samantekt (2-3 málsgreinar)	Hvað lausnin gerir, tilgangur hennar, og hlutverk í stærra samhengi
Tengill í ítarlegri skjölun	Sjá [ítarlegri skjölun](Docs/INDEX.md) fyrir frekari upplýsingar.
Tæknilýsing	Forritunarmál, framework útgáfur, helstu tæknival
Yfirlitsmynd	Hvernig helstu hlutar lausnarinnar vinna saman (mynd eða stutt lýsing)
Ytri tengsl	Hvaða ytri tengingar, gagnagrunnar eða þjónustur lausnin stólar á. Secrets og VPN ef við á.
Frávik	Öll helstu frávik frá þróunarhandbók, skjalfest og rökstudd
Uppsetning fyrir forritara	Hvernig nýr forritari kemur lausninni upp á sinni vél og keyrir hana
Gagnagrunnur (ef við á)	Uppsetning, SQL scriptur, helstu notendur, viðhald

18.4 Kröfur um uppsetningarleiðbeiningar

Uppsetningarleiðbeiningar skulu vera nægilega ítarlegar til að **reyndur forritari** sem hefur aldrei séð lausnina geti komið henni í gang án utanaðkomandi aðstoðar.

Þær skulu ná til:

- þróunarumhverfis og kröfur til þess (t.d. .NET útgáfa, Docker, tól),
- ytri þjónustu sem þarf að herma (mock) eða tengjast,
- stillingaskráa, environment breyta, secrets og aðgangs,
- hvernig á að nálgast logga í þróunarumhverfi,
- hvernig á að keyra sjálfvirkar prófanir á vélinni.

18.5 Skjölun bakgrunnsvinnslna (CronJobs)

Ef lausn inniheldur CronJob stillingar (t.d. í .kube/ möppu) skal sérstakt skjal (Docs/cronjobs.md) vera til staðar sem inniheldur:

Þáttur	Lýsing
Yfirlitstafia	Öll cron job, tímaáætlanir þeirra og CLI breytur sem notaðar eru
Samhliða keyrsla	Mat á hvort samhliða keyrsla sama jobs sé örugg
Bilanaviðbrögð	Hvernig stjórnendur skulu bregðast við ef keyrsla mistekst
Tímaárekstrar	Viðvörun ef áætlanir geta skarast eða valdið vandamálum

18.6 Skjölun í forritunarkóða

Skjölun í sjálfum forritunarkóðanum fellur undir kafla *Forritunarkóði og vinnureglur*. Eftirfarandi atriði eru hluti af gæðamati við skil:

Krafa	Lýsing
Inline XML summary	Allir public klasar, enums, föll, eiginleikar (properties) og svæði (fields) skulu hafa XML summary skjölun
Hágæða athugasemdir	Athugasemdir skulu lýsa *af hverju* kóðinn gerir það sem hann gerir, ekki *hvernig*
Frávik skjalfest	Frávik frá stöðluðum venjum skulu vera skjalfest og rökstudd í kóða
Vantar skjölun = athugasemd í rýni	Ef skjölun vantar eða er ófullnægjandi kemur það fram sem athugasemd í yfirferðarskjali

XML Summary dæmi

```
/// <summary>
/// Validates that the patient has an active treatment relationship
/// with the requesting clinician before granting access to clinical data.
/// Required by healthcare data access regulations.
/// </summary>
/// <param name="patientId">National ID of the patient.</param>
/// <param name="clinicianId">National ID of the requesting clinician.</param>
/// <returns>True if an active treatment relationship exists; otherwise false.</returns>
/// <exception cref="UnauthorizedAccessException">
/// Thrown when the clinician's credentials cannot be verified.
/// </exception>
public async Task<bool> ValidateTreatmentRelationship(string patientId, string clinicianId)
```

18.7 Gæðamat á skjölun við skil

Eftirfarandi atriði eru metin sjálfvirkt og/eða handvirkt við hverja breytingabeiðni (PR):

Matskrafa	**Lýsing**
Skjölun endurspeglar breytingar	Ef PR breytir virkni eða arkitektúr skulu README og /Docs skjöl vera uppfærð samhliða
README til staðar og uppfært	README.md í rót skal vera til staðar og endurspegla núverandi stöðu
Tengill í /Docs	README skal innihalda tengil í Docs/INDEX.md
Nægjanleg fyrir reyndan forritara	Reyndur forritari getur komið sér í gang án utanaðkomandi aðstoðar
Nægjanleg fyrir verkefnastjóra og rekstur	Tilgangur, arkitektúr og helstu tengsl eru skiljanleg án kóðalestur
Frávik skjalfest	Öll frávik frá stöðlum og þróunarhandbók eru skjalfest og rökstudd
Athugasemdir í kóða	Hágæða athugasemdir og XML summary á public einingum

Ef skjölun uppfyllir ekki þessar kröfur kemur það fram sem athugasemd í yfirferðarskjali kóðarýni (sjá viðauka *Yfirferðarskjal kóðarýni*), almennt flokkuð sem **Org-Policy** eða **Best-Practice** eftir alvarleika.

19. Viðauki – Docker kröfur og bestu venjur

Þessi viðauki lýsir nákvæmum kröfum til Docker eininga sem afhentar eru verkkaupa. Viðaukinn útfærir þær reglur sem settar eru fram í kafla *Útgáfuferli og keyrslustýring* og er ætlaður forriturum og DevOps sérfræðingum verksala.

19.1 Staðsetning og uppbygging

Krafa	**Lýsing**
Staðsetning Dockerfile	Í rót verkefnis
Fjöldi eininga	Ein Docker eining per sjálfstæða keyrslueiningu
Skjölun	Dockerfile skal innihalda skýrar athugasemdir á ensku
Læsileiki	Dockerfile skal vera einföld, læsileg og fyrirsjáanleg

19.2 Build ferli

Krafa	**Lýsing**
Build/compile í Docker	**Óheimilt** — Docker eining pakkar einungis inn forsmíðuðum kóða
Prófanir í Docker	**Óheimilt** — prófanir keyrðar utan Docker eininga, í CI pípulínum
Endurbygging eftir CI	**Óheimilt** — Docker má ekki endurbyggja kóða eftir að prófanir hafa verið keyrðar
Multi-stage builds	**Heimilt** og æskilegt þegar við á

19.3 Keyrsla og öryggi

Krafa	**Lýsing**
Root notandi	Docker einingar mega **ekki** keyra hugbúnað sem root
Notendaréttindi	Dockerfile skal stofna non-root notanda og keyra hugbúnað undir heimildum hans
Árásarflötur	Leitast skal við að lágmarka innihald eininga og óþarfa pakka
Alpine einingar	Nota Alpine grunn einingar þegar tæknilega og rekstrarlega skynsamlegt

19.4 Stærð og endurtekningarhæfni

Krafa	**Lýsing**
Stærð	Eins litlar og mögulegt er, fyrir hraðari build og útgáfu
Endurtekning	Smíði Docker einingar skal vera endurtakanleg og óháð umhverfi
Fyrirsjáanleiki	Sama Docker eining skal hegða sér eins í öllum umhverfum

19.5 Logging og skráakerfi

Krafa	**Lýsing**
Logging	Allar logfærslur í stdout eða stderr
Logskrár á disk	**Óheimilt**
Logasöfnun	Meðhöndluð af Kubernetes og rekstrarinnviðum verkkaupa

19.6 Stillingar og leyndarmál

Krafa	Lýsing
Leyndarmál í Docker einingum	**Óheimilt**
Appsettings / env skrár með viðkvæmum gögnum	**Óheimilt**
Tengistrengir eða IP tölur í Dockerfile	**Óheimilt**
Leyndarmál	Geymd í Vault; sprautuð inn við keyrslu í Kubernetes
Birting í loggum	**Óheimilt** — leyndarmál mega aldrei birtast í loggum eða með env skipun

19.7 Netkerfi og port

Krafa	Lýsing
Opin port	Dockerfile skal skilgreina nákvæmlega hvaða port eru opin
Fjöldi porta	Halda í lágmarki
Docker networking	Ef notað, skal fylgja viðeigandi skjölun

19.8 Merking (tagging)

Krafa	Lýsing
Útgáfunúmer	Docker einingar merktar bæði með útgáfunúmeri og git commit SHA
Föst tög (latest-dev)	Einungis fyrir test umhverfi
Föst tög í PreProd/Prod	**Óheimilt** — einungis CalVer tög
imagePullPolicy	Kubernetes poddar stilltir með imagePullPolicy: Always

19.9 Startup og README

Krafa	Lýsing
README	Skal taka fram hvað þarf til að keyra upp Docker einingu
Build leiðbeiningar	Hvernig á að byggja einingu
Keyrsluleiðbeiningar	Dæmi um command line keyrslu

19.10 Skýrt bann (samantekt)

Atriði	Staða
Docker-in-Docker (DinD)	**Óheimilt**
Build í runtime einingu	**Óheimilt**
Prófanir í Docker einingu	**Óheimilt**
Leyndarmál í Dockerfile/einingu	**Óheimilt**
Root keyrsla	**Óheimilt**

20. Viðauki – Kubernetes kröfur og samhæfni

Þessi viðauki lýsir nákvæmum kröfum til Kubernetes-samhæfni hugbúnaðarlausna sem afhentar eru verkkaupa. Viðaukinn útfærir þær reglur sem settar eru fram í kafla *Útgáfuferli og keyrslustýring* og er ætlaður forriturum og DevOps sérfræðingum verksala.

Rekstur Kubernetes innviða, Argo CD stillingar, vöktun, afritun og neyðarferlar eru skilgreind í *Rekstrarhandbók verkkaupa*.

20.1 Almennt

Krafa	Lýsing
Keyrsluumhverfi	Allar lausnir (að undanskildum gagnagrunnum) skulu geta keyrt í Kubernetes
Rekstur	Kubernetes innviðir eru hýstir og reknir af verkkaupa
Ábyrgð verksala	Hugbúnaður skal vera Kubernetes-hæfur
Ephemeral umhverfi	Ekki studd vegna flækjustigs innviða

20.2 GitOps og Argo CD samhæfni

Krafa	Lýsing
GitOps	Lausnir skulu vera samhæfðar GitOps útgáfumódeli
Argo CD	Notað til útgáfu og samræmingar af verkkaupa
Innviðakóði	Kubernetes stillingar geymdar með kóðanum í .kube/ möppu
Möppustrúktúr	Uppbygging .kube/ skal endurspegla Argo CD repository skipulag

Nákvæm uppsetning Argo CD, sync-stefnur og samþykktir eru skilgreindar í Rekstrarhandbók.

20.3 Heilsuathuganir (Probes)

Allar þjónustur skulu bjóða upp á liveness og readiness probe til að styðja sjálfvirka endurræsingu og umferðastýringu.

Probe	Endapunktur	Hlutverk	Afleiðing ef bregst
Liveness	/system/alive	Metur hvort þjónustan sé í gangi og hafi ekki hrunið eða sé í lás	Kubernetes endurræsir þjónustuna sjálfkrafa
Readiness	/system/ready	Metur hvort þjónustan sé tilbúin til að taka við beiðnum; prófar tengingar við nauðsynlegar ytri einingar	Kubernetes stöðvar umferð að þjónustunni en lætur hana keyra áfram

20.3.1 Kröfur um útfærslu

Krafa	Lýsing
Staðlaðar forritunareiningar	Verksala er skylt að nýta staðlaðar forritunareiningar verkkaupa fyrir heilsuathuganir (sjá viðauka *Stöðluð forritunarstöfn*)
Raunveruleg staða	Probes skulu endurspegla raunverulega keyrslustöðu, ekki skila föstum gildum
Annað port (valkvætt)	Heimilt er að birta /system/* endapunkta á porti 9102
Engin ytri aðgengi	/system/* endapunktar skulu ekki vera aðgengilegir af internetinu

Lausnir án probes eru ekki útgáfuhæfar.

20.4 Logging og mælingar

Krafa	Lýsing
Logging	Allar logfærslur í stdout/stderr
Logform	Structured logging skylt (JSON/ECS)
Logskrár	Skrif í logskrár á disk **óheimilt**
Metrics	Lausnir skulu gera tæknilegar metrics aðgengilegar
Söfnun	Á ábyrgð Kubernetes innviða verkkaupa

Sjá kafla „Rekstur og eftirlit" fyrir nákvæmar kröfur um logging, metrics og útgáfuupplýsingar.

20.5 Stillingar og keyrsluleyndarmál

Krafa	Lýsing
Stillingar	Lausnir skulu lesa stillingar úr keyrsluumhverfi
Secrets	Öll keyrsluleyndarmál í Vault
Innsprautun	Secrets sprautuð inn við keyrslu
Geymsla í kóða	**Óheimilt** — secrets mega ekki vera í kóða, Docker einingum eða config skrám
Birting	Secrets mega ekki birtast í loggum eða debug úttaki

Sjá viðauka „Geymsla leyndarmála" fyrir .NET appsettings högun og Vault innsprautunarmynstur.

20.6 Keyrslueiginleikar og álagsþol

Krafa	Lýsing
Endurræsing	Lausnir skulu þola endurræsingu tilvika án gagnataps
Skölun	Lausnir skulu styðja lárétta skölun (load balancing milli tilvika)
Stöðuleysi	Lausnir skulu vera stateless
Ástand	Lausnir mega ekki treysta á staðbundið ástand (in-memory state milli beiðna)

20.7 Bann og takmarkanir

Atriði	Staða
Skrif á disk í runtime	**Óheimilt** (nema tímabundið, t.d. /tmp)
Handvirk stilling í keyrslu	**Óheimilt**

Sérsniðin Kubernetes hegðun án samþykkis	**Óheimilt**
Beinar breytingar í cluster af verksala	**Óheimilt**

21. Viðauki – Skipulag .agent/ möppu

Þessi viðauki lýsir skipulagi, innihaldi og tilgangi .agent/ möppunnar sem fylgir öllum kóðageymslum verkkaupa. Möppan veitir gervigreindarverkfærum skipulegt samhengi, staðla og leiðbeiningar sem stuðla að gæðum og samræmi í gervigreindarstuddri hugbúnaðarþróun.

21.1 Uppruni og staðall

.agent/ möppan byggir á opna staðlinum **Agent OS v3.0** sem er þróaður af Builder Methods og aðgengilegur á:

- Vefur: https://buildermethods.com/agent-os
- GitHub: https://github.com/buildermethods/agent-os
- Skjölun: https://buildermethods.com/agent-os/file-structure

Verkkaupi hefur innleitt Agent OS staðalinn í sínar sniðmátskóðageymslur (template repositories) með einu fráviki: möppuheitið er .agent/ í stað agent-os/ (falin mappa með punkti). Innra skipulag, skráarsnið og YAML-skilgreiningar eru eins og upprunalegi staðallinn.

21.2 Möppuskipulag

```
.agent/
├── config.yml                  # Útgáfa staðals og sjálfgefið snið (profile)
├── README.md                   # Stefna verkkaupa um gervigreindarstuddur þróun
├── product/                    # Afurðaskjölun
│   ├── mission.md              # Tilgangur og markmið kóðageymslu
│   ├── roadmap.md              # Áætlun um virkni og þróunarstig
│   └── tech-stack.md           # Tæknistakkur, skylduforritunarsöfn, takmarkanir
├── standards/                  # Kóðastaðlar eftir flokkum
│   ├── index.yml               # Véllesanlegt yfirlit yfir alla staðla
│   ├── {flokkur}/              # Ein undirmappa per flokk (t.d. api/, database/,
ui/)
│   │   └── {staðall}.md        # Einstakur staðalskjal
│   └── ...
└── specs/                      # Frátekið fyrir mótaðar virknisskilgreiningar
    └── {YYYY-MM-DD-HHMM-feature-slug}/
        ├── plan.md
        ├── shape.md
        ├── standards.md
        ├── references.md
        └── visuals/
```

21.3 Skráarlýsingar

21.3.1 README.md

Sameiginlegt stefnuskjal verkkaupa um gervigreindarstuddur þróun. Inniheldur:

- Afstöðu verkkaupa til notkunar gervigreindar í þróun
- Takmarkanir vegna heilbrigðisgagna (engin raunveruleg sjúklingagögn, engar sjálfstæðar klínískar ákvarðanir)
- Tilvísun í þróunarhandbók verkkaupa
- Athugasemd um EU AI Act / GDPR

Þetta skjal er eins í öllum kóðageymslum verkkaupa.

21.3.2 product/ (3 skrár)

Skrá	Innihald	Sameiginlegt eða sértækt?
mission.md	Tilgangur kóðageymslunnar, meginreglur, tilvísun í viðmiðunarútfærslur	Sértækt eftir tegund lausnar

roadmap.md	Þróunaráætlun í áföngum (Phase 1, Phase 2 o.s.frv.)	Sértækt eftir tegund lausnar
tech-stack.md	Tæknistakkur, skyldupakkar, arkitektúrtakmarkanir, óheimilar tæknilausnir	Sértækt eftir tegund lausnar

21.3.3 standards/ (breytilegt fjöldi skráa)

Staðlar eru skipulagðir í undirmöppur eftir flokkum. Hver möppuheiti samsvarar tæknilegum flokki (t.d. api/, database/, ui/, security/). Innan hvers flokks er ein eða fleiri Markdown-skrá sem lýsir einum staðli.

index.yml í rót standards/ möppunnar er véllesanlegt yfirlit:

```
flokkur:
  staðall-nafn:
    description: Eins setnings lýsing á staðlinum
```

21.3.4 Stöðluð flokkaheiti eftir tegund lausnar

Flokkur	Bakendi	Framendi	NuGet safn
architecture/	✓	✓	✓
api/	✓		
ui/		✓	
authentication/		✓	
database/	✓	✓	✓
dependency-injection/	✓	✓	✓
logging/	✓	✓	
middleware/	✓	✓	
deployment/	✓	✓	
code-quality/	✓	✓	✓
testing/	✓	✓	✓
security/	✓	✓	✓
background-jobs/	✓		
packaging/			✓
ci-cd/			✓

21.3.5 specs/ (frátekið)

Frátekið fyrir mótaðar virknisskilgreiningar (shaped feature specifications) samkvæmt Agent OS staðlinum. Ekki enn í notkun hjá verkkaupa en geymt til framtíðarnotkunar.

21.4 Snið einstakra staðlaskráa

Staðlaskrár í standards/ fylgja samræmdu sniði:

```
# {Heiti staðals}

{Ein setning sem lýsir staðlinum og meginreglunni.}

## {Kaflafyrirsögn}
{Útskýring eða kóðadæmi}

## Rules
- {Skipanir: alltaf gera X, aldrei gera Y}
```

Meginreglur um innihald staðlaskráa:

Meginregla	Lýsing
Reglan fyrst	Segja hvað á að gera áður en útskýrt er hvers vegna
Kóðadæmi	Sýna fremur en segja (C#, Razor, SQL, YAML eftir því sem við á)
Hnitmiðað	Lágmarka orð — hvert orð kostar tóka í gervigreindarsamhengi
Einn staðall per hugtak	Ekki blanda óskyld mynstur í sömu skrá
Rules kafli	Flestar skrár enda á hnitmiðuðum lista yfir skipanir

21.5 Kröfur til verksala

Krafa	Lýsing
Virða leiðbeiningar	Gervigreindarverkfæri verksala skulu vera stillt þannig að þau lesi og virði .agent/ möppuna
Viðhalda og auðga	Verksali skal uppfæra .agent/ möppuna jafnóðum og þróun miðar áfram — ekki einungis við skil
Bæta við staðlum ef þörf	Ef lausnin krefst sértækra staðla sem ekki eru til staðar í sniðmátinu skal verksali bæta þeim við í réttum flokki
Bæta við product/ skjölun	Verksali skal uppfæra mission.md, roadmap.md og tech-stack.md þannig að þau endurspegli raunverulega stöðu lausnar
Nota specs/ ef við á	Ef verkefni notar mótaðar virknisskilgreiningar skulu þær vera skjalfestar í specs/ möppunni
Hluti af rýni	Innihald og gæði .agent/ möppu eru hluti af gæðamati við skil og verða metin í kóðarýni

21.6 Hvaða skrár verksali má ekki breyta

Eftirfarandi skrár eru í umsjón verkkaupa og verksali skal ekki breyta þeim nema í samráði:

Skrá	Ástæða
config.yml	Útgáfa staðals ákveðin af verkkaupa
README.md	Stefna verkkaupa — sameiginleg í öllum kóðageymslum

21.7 Tengsl við sniðmátskóðageymslur

Verkkaupi viðheldur þremur sniðmátskóðageymslum sem innihalda fullskilgreinda .agent/ möppu:

Sniðmát	Tegund lausnar	Fjöldi staðlaskráa
template-repo-dotnet-backend	Bakendaþjónustur og API	~17
template-repo-dotnet-frontend	Framendalausnir (SSR)	~15
template-repo-dotnet-nugetlibrary	NuGet forritunarsöfn	~8

Þegar ný kóðageymsla er stofnuð af verkkaupa fylgir viðeigandi .agent/ mappa sjálfkrafa.

Sjá viðauka „Þjónustur og verkfæri verkkaupa" fyrir aðgang að sniðmátskóðageymslum og lausnarbanka.

22. Viðauki – Gervigreindarlausnir – arkitektúr og kröfur

Þessi viðauki lýsir ítarlegum tæknilegum kröfum til gervigreindarlausna (AI agents, MCP) sem verða hluti af keyrsluafurð verkkaupa. Viðaukinn á við þegar verkefni fela í sér gervigreindarvirkni í afurð og útfærir þær reglur sem settar eru fram í kafla 8B.

22.1 Gervigreindaragentar sem bakendaþjónustur

Gervigreindaragentar skulu aldrei vera keyrðir í framenda né sem stakar, ósjálfstæðar forritunarkriptur. Agentar skulu:

- vera útfærðir sem ASP.NET bakendaþjónustur og fylgja sömu lagskiptingu og önnur bakenda-API eða þjónustur,
- vera keyrðir í Docker og Kubernetes og vera útgefnir í gegnum sama CI/CD ferli og aðrar þjónustur.

22.2 Arkitektúr og högun

Gervigreindarlausnir skulu fylgja sömu arkitektúrviðmiðum og skilgreind eru í kafla 9 (Hönnun og arkitektúr), m.a.:

- SOLID + Separation of Concerns
- Onion / Multilayered architecture
- Vertical slice þar sem við á

- REPR-högun fyrir API

22.2.1 Dæmigerð lagskipting

Lag	Hlutverk	Leyfilegt innihald	Óleyfilegt innihald
Controllers / Endpoints	Inntak/úttak og HTTP samningar	DTO möppun, auth/validation hooks	Kvaðningar, LLM-köll, flæðistjórnun
Orchestration / Managers	Stýrir flæði og reglum	Flæðisreglur, fallback, villumeðhöndlun	Beinn aðgangur að viðmóti eða framenda
Agent Layer	Agent lifecycle og samtal	Skilgreiningar samþykkts forritunarsafns, kvaðningar og ræsing	Gagnagrunnslogík, óstýrt minni gervigreindar
Infrastructure	Hjúpun ytri þjónusta	LLM client, vector store client, kvaðninga sniðmát	Viðskiptareglur
Repositories	Gögn og aðgangsstýring	DB/FHIR/gagnasöfn	Kvaðningar- eða agent-stýring

22.3 MCP – Model, Message og Control

Allar agent-lausnir skulu skilgreina skýrt MCP-högun sína:

MCP hluti	Lágmarkskrafa	Dæmi um sönnun (deliverable)
Model	Skilgreina og tilgreina líkan og útgáfu þess sem notað er, stillingar á líkani, fallback og heimildalisti (allowed list)	model-policy.md + config í repo
Message	Skilgreina inntak/úttak sem formlega samninga (JSON schemas)	OpenAPI + JSON schema + DTOs
Control	Skilgreina hver stjórnar flæði, stopp-punkta (stop-points) agenta, ákvarðanir, ábyrgð og audit	Sequence diagram + audit event list

22.4 Öryggi og persónuvernd

Gervigreindarlausnir falla undir sömu öryggiskröfur og önnur bakendakerfi (sjá kafla 12), auk eftirfarandi sérkrafna:

- Engin heilbrigðisgögn mega fara í LLM gervigreindarmódel nema samþykkt liggi fyrir, gögn séu lágmörkuð og rekjanleiki tryggður.
- Kvaðningar og svör skulu logguð á öruggan hátt.
- Allar agent-aðgerðir skulu vera rekjanlegar (audit trail).

Agentar mega aldrei:

- geyma gögn í sameiginlegu minni (memory) án aðgangsstýringar,
- deila gögnum eða niðurstöðu kvaðninga milli notenda,
- taka sjálfstæðar ákvarðanir með réttaráhrifum nema sérstaklega skilgreint.

22.5 Prófanir og gæðastýring

Gervigreindarlausnir skulu vera prófanlegar án LLM.

Prófunartegund	Krafa	Mælanlegt viðmið
Einingapróf	Uppbygging kvaðninga, flæðistjórnun og villumeðhöndlun	Lágmarks coverage skilgreint í verkefni
Samþættingarpróf	Hermd LLM svör, köll í ytri tól og kerfi	Keyra í CI
Endurtekningarhæfni	Deterministic niðurstöður án raun-LLM	Mock mode / test harness
Reglupróf	Bannreglur (t.d. "aldrei senda X")	Neikvæð prófun

Allir agentar skulu skrá inntak (með hreinsun), ákvarðanir og villur í keyrslu samkvæmt leiðbeiningum um structured logging (sjá kafla 19).

22.6 Viðmiðatafla

Viðmið	Æskileg nálgun	Óæskileg nálgun
Fjöldi agenta	Fáir, afmarkaðir agentar með skýrt hlutverk	Margir, tengdir eða sjálfstýrðir agentar
Ábyrgð	Skýr ábyrgð í bakenda (orchestration)	Ákvarðanir teknar inni í LLM

Flæðisstýring	Stýrt flæði skilgreint í kóða	Ófyrirsjáanleg agent-samskipti
Tæknistakkur	ASP.NET + samþykkt Microsoft forritunarsafn	Sérlausnir eða heimasmíðaðar lausnir
Samþættingar	Í gegnum staðlað bakenda API	Beinar tengingar frá agent
Kvaðningar	Endurnýtanlegar og útgáfustýrðar	Harðkóðaðar eða ósýnilegar kvaðningar
Fjöl-agent kerfi	Ein-agent lausn þar sem mögulegt	Flókin multi-agent kerfi
LLM háð	LLM sem útfærsla, ekki kjarni	Lausn ókeyranleg án LLM
Viðhald	Fyrirsjáanlegt, fáir hreyfanlegir hlutar	Mörg sérsniðin módel og flæði
Rekstur	Samræmd logging, metrics og CI/CD	Sérlausnir fyrir agenta

23. Viðauki – Æskilegt gervigreindarþróunarferli

Þessi viðauki lýsir æskilegu vinnulagi við gervigreindarstudda hugbúnaðarþróun fyrir verkkaupa. Vinnulagið er samsett af fimm áföngum sem byggja á þeirri grunnreglu að *áætlanagerð og útfærsluáætlun séu sjálfstæðir verkáfangar sem ljúka skal — og rýna skal — áður en gervigreindarstudd kóðasmíði hefst.*

Viðaukinn útfærir væntingar verkkaupa sem settar eru fram í kafla 11.1 og á við bæði um:

- **Forskrift (Seeding)** — þar sem verkkaupi nýtir gervigreindarverkfæri til að búa til tæknilega forskrift út frá hönnunar- og greiningargögnum (sjá Mynd 1 í kafla A).
- **Verkefnavinnu (Project execution)** — þar sem verksali nýtir sömu aðferðafræði til að ljúka útfærslu lausnar.

Í þessum viðauka er talað um *útfærsluaðila* — þ.e. þann aðila sem ber ábyrgð á útfærslu hverju sinni (verkkaupi í Forskrift, verksali í Verkefnavinnu).

23.1 Tilgangur og gildissvið

Atriði	**Lýsing**
Tilgangur	Tryggja að gervigreindarstudd kóðasmíði byggi á rýndri og samþykktri áætlun, en sé ekki ómarkvist viðbragð við stökum verkbeiðnum
Gildissvið	Öll verkefni þar sem gervigreindarverkfæri eru notuð sem aðalsmíðatæki. Bindandi fyrir Leið B framendaverkefni (kafli 8.2.2), æskilegt fyrir öll önnur verkefni
Útfærsluaðili	Verkkaupi (Forskrift) eða verksali (Verkefnavinna) eftir áfanga ferilsins
Rýniaðili	Tæknilegur ábyrgðaraðili verkkaupa (SH)
Vinnsluskjöl	Geymd í -docs kóðageymslu verkefnisins, númeruð frá 01

23.2 Yfirlit yfir ferlið

#	**Áfangi**	**Útfærsluaðili**	**Niðurstaða**	**Gæðahlið**
1	Áætlanagerð (*Planning*)	Útfærsluaðili	Númeruð útfærsluáætlunarskjöl í -docs kóðageymslu	Innra sjálfsmat á þekju
2	Framkvæmdaáætlun (*Execution Plan*)	Útfærsluaðili	Vinnulotuskipt framkvæmdaáætlun með afritunarhæfum kvaðningum	Innra sjálfsmat á yfirgripi og að allir verkþættir séu til staðar
3	Skuldbinding og gap-rýni	Útfærsluaðili → SH	Samþykkt framkvæmdaáætlun	**SKAL** — engin útfærsla hefst fyrr en SH samþykkir

4	Útfærsla session prompta	Útfærsluaðili	Smíðuð virkni samkvæmt framkvæmdaáætlun, sannvottuð í hverri vinnulotu	Sannvottun í lok vinnulotu
5	Lokastaðfesting og prófanir	Útfærsluaðili	Tilbúin lausn, full skjalað í -docs og .agent möppum	Afhending til verkkaupa

Hverjum áfanga er lýst nánar hér á eftir.

23.3 Áfangi 1: Áætlanagerð (Planning Phase)

Markmið áfangans er að umbreyta dreifðum greiningar- og hönnunargögnum í eitt samhæft sett af útfærsluáætlunarskjölum sem ná yfir allar virknieiningar lausnarinnar.

23.3.1 Inntak í áætlanagerð

Útfærsluaðili skal safna eftirfarandi heimildum og gera þær aðgengilegar gervigreindarvélinni á einum stað, sérfræðingar SH munu leggja til sérstaka -docs github kóðageymslu sem ætluð er fyrir öll undirbúnings og skipulagsskjöl verkefnisins:

Flokkur	**Dæmi um heimildir**
Útboðsgögn	Útboðsskilmálar, kröfutexti, Þ-viðauki, samningsdrög
Greining	Notendasögur, kröfugreining, viðtalsskýrslur, fundargerðir, klínískar kröfur
Núverandi kerfi	Greining á núverandi lausnum, kortlagning samtenginga, ytri og innri gagnasöfn
Hönnun	Hönnunarstaðall, Figma skjámyndir, viðmótsstaðall, hönnunarkerfi
Gögn og líkön	Gagnagrunnshögun, yfirlitsmyndir, FHIR samræming
Handbækur SH	Þróunarhandbók, Tæknistefna, Rekstrarhandbók, Hönnunarhandbók
Sniðmát	.agent/ mappa kóðageymslu (sjá viðauka „Skipulag .agent/ möppu")

23.3.2 Kröfur til gervigreindarvélar

Áfangi 1 krefst gervigreindarvélar sem ræður við *langtíma (long-horizon) ósamfellda áætlanagerð* — þ.e. getur lesið og samhæft tugi til hundruð skjala án mannlegrar íhlutunar í hverju skrefi.

Krafa	**Lýsing**
Ítarleg áætlanagerð (long context)	Verkfærið skal geta unnið sjálfstætt í langan tíma (klukkustundum) án mannlegrar íhlutunar
Heildstæð skjalalestur	Verkfærið skal lesa og vinna með heildar skjalasafn, ekki einungis valda útdrætti
Skrifað út í git	Verkfærið skal skrifa niðurstöður beint í -docs kóðageymslu
Ítranir	Verkfærið skal styðja að áætlanir og verklag sé ítrað og betrumbætt í samtali við útfærsluaðila

Dæmi um samþykkt verkfæri (apríl 2026): Devin.AI. Verkkaupi viðurkennir einnig önnur sambærileg agent-verkfæri sem uppfylla kröfurnar hér að ofan. Notkun verkfæranna fellur undir sömu öryggis- og gagnameðhöndlunarkröfur og lýst er í kafla 11 (gagnaöryggi) og kafla 11 (upplýsingaskylda).

23.3.3 Útkoma: númeruð útfærsluáætlunarskjöl

Niðurstöður Áfanga 1 skulu skrifaðar í -docs kóðageymslu verkefnisins undir möppunni /docs/plans/ í markdown sniði og númeraðar frá 01 (sjá dæmi hér að neðan):

```
-docs/
└── docs/
    └── plans/
        ├── 00-overview.md           # Yfirlit yfir öll áætlunarskjöl
        ├── 00-coverage-matrix.md    # Þekjugrein á SKAL-kröfum útboðs
```

```
        ├── 01-patient-onboarding.md
        ├── 02-clinician-workflows.md
        ├── 03-fhir-integrations.md
        ├── 04-audit-and-logging.md
        └── ...
```

Hvert áætlunarskjal skal innihalda eftirfarandi að lágmarki (versali er hvattur til að útfæra öll þau viðbótar áætlunarskjöl sem gervigreindarvélin telur rétt að bæta við):

Hluti	**Lýsing**
Heiti og virknisvæði	Skýrt heiti og afmörkun
Heimildatilvísanir	Hvaða inntaksskjöl liggja til grundvallar hverrar kröfu
Virknislýsing	Hvað á að gera, frá sjónarhóli notanda
Viðtökuviðmið	Mælanleg viðmið um hvenær verkið telst lokið
Tæknilegir íhlutir	Hvaða þjónustur, gagnagrunnar, viðmótshlutar o.þ.h. koma við sögu
Samhæfniskröfur	EU AI Act, lækningatæki, GDPR, aðgengi (eftir því sem við á)
Tengsl við ytri kerfi	Háðir og tengsl við aðrar virknieiningar og kerfi
Flækjustigsmat	S/M/L mat fyrir áætlun

23.3.4 Ítranir og rýni

Útfærsluaðili skal *ekki* samþykkja fyrstu útgáfu áætlunarinnar í blindni. Skylt er að ítra a.m.k. einu sinni með gervigreindarvélinni (hvatt er til að nota aðra vél til ítrunar) og keyra sjálfsmat á þekju gagnvart útboðskröfum.

Viðmið við sjálfsmat	**Spurning sem á að svara**
Þekja á SKAL-kröfum	Er hver SKAL-krafa útboðs rakin í a.m.k. einu áætlunarskjali?
Þekja á virknisvæðum	Eru öll virknisvæði í greiningu innifalin í einhverju af áætlunarskjölunum?
Þekja á samhæfniskröfum	Eru EU AI Act, lækningatæki og GDPR sjónarmið tekin til hliðsjónar þar sem við á?
Tæknileg samkvæmni	Er einhver áætlunarliður í andstöðu við Tæknistefnu eða kafla 5 í þessari handbók? Þarf undanþágur?
Innra samhengi	Eru áætlunarskjölin innbyrðis samkvæm (engar tvíverkanir, engar gloppur eða göt í útfærslum)?

23.3.5 Dæmi um kvaðningar (Phase 1 prompts)

Eftirfarandi kvaðningar eru ætlaðar sem leiðbeinandi fyrir þróunarteymi eða til beinna afnota. Þær eru á ensku því gervigreindarverkfæri skila almennt betri niðurstöðu á ensku. Einnig skal allur kóði og skjölun vera á ensku skv. kafla 10.

Initial planning prompt:

```
You are a senior software architect creating an implementation plan for a
software project governed by the buyer's development handbook.

INPUT DOCUMENTS:
Read every document in these directories thoroughly. Do not skip any file.
- /docs/plans/        Existing plans (if any) — understand current state
- /tendering/         Tendering requirements, SHALL/SHOULD specifications
- /analysis/          Requirements analysis, user stories, meeting notes, ADRs
- /requirements/      Domain-specific requirements and SME interview notes
```

```
- /design/            Design system, Figma exports, UI guidelines
- /database/          Database design, ER diagrams, data mappings
- /handbooks/         Þróunarhandbók, Tæknistefna, Rekstrarhandbók, Hönnunarhandbók
- .agent/standards/   Coding standards and conventions for this project

If any of these directories do not exist, skip them silently.

YOUR TASK:
Produce numbered implementation plan documents — one per functional domain —
written to /docs/plans/ in markdown. Number files starting from 01.

EACH PLAN DOCUMENT MUST CONTAIN:

1. **Title and scope** — feature name, one-paragraph summary of what it covers.

2. **Source references** — table of input documents that drive this plan, with
   section/page numbers where applicable.

3. **Key design decisions** — numbered list of all design questions that arose,
   with the chosen resolution and rationale. Mark each as RESOLVED or OPEN.
   All decisions must be resolved before Phase 2 can begin.

4. **Architecture** — technical design including:
   - Database schema (exact DDL where applicable)
   - API endpoint specifications (routes, methods, request/response shapes)
   - Service/manager/repository class responsibilities
   - Sequence diagrams for complex flows (Mermaid syntax)
   - Configuration and environment variables

5. **Security analysis** — authentication, authorization, data protection,
   OWASP considerations, and applicable regulatory constraints (e.g., GDPR,
   EU AI Act, sector-specific rules).

6. **Functional specification** — user-visible behaviour, acceptance criteria,
   edge cases, error handling.

7. **Technical components** — which services, databases, UI components,
   integrations, packages/libraries, and external systems are affected.

8. **Dependencies** — which other plan documents this plan depends on or
   affects.

9. **Complexity estimate** — S / M / L with brief justification.

ALSO PRODUCE:
- /docs/plans/00-overview.md — lists every plan file with a one-sentence
```

```
  summary and an overall dependency diagram in Mermaid syntax.
- /docs/plans/00-coverage-matrix.md — maps every SHALL/SKAL requirement from
  /tendering/ and /handbooks/ to the plan file(s) that cover it. Flag any
  gaps.

HARD CONSTRAINTS (do not violate):
- All code, comments, and plan documents must be in English.
- Follow the technology choices mandated by the project's Tæknistefna and
  Þróunarhandbók. Do NOT introduce technologies not approved in those
  documents.
- Do NOT leave design decisions OPEN without flagging them prominently.
  The execution plan (Phase 2) cannot begin until all decisions are RESOLVED.

ITERATION:
After producing the first draft, self-review against these criteria:
- Is every SHALL/SKAL requirement covered in at least one plan?
- Are all plans internally consistent (no contradictions, no gaps)?
- Are compliance constraints (regulatory, privacy, accessibility) addressed
  where applicable?
- Is anything in conflict with the project's handbooks?
List any issues found and update the plans accordingly.

When done, summarise coverage by feature area and list any source documents
you were unable to fully reconcile..
```

23.3.6 Coverage iteration prompt:

```
You have produced implementation plan documents in /docs/plans/. Before these
plans can proceed to Phase 2 (execution planning), they must pass a structured
quality gate. Perform the following checks and fix any gaps found.
STEP 1 — REQUIREMENT COVERAGE MATRIX
Review every SKAL/SHALL requirement in /tendering/ and every mandatory
requirement in /handbooks/. For each requirement:
- Identify which plan file(s) cover it
- Assess whether the coverage is complete or partial
Produce /docs/plans/00-coverage-matrix.md with these columns:
| Req ID | Source Document | Section | Requirement Summary | Plan File(s) | Coverage |
Gap Description |
|--------|----------------|---------|---------------------|--------------|----------|-
----------------|
Coverage values: "Full", "Partial — [what's missing]", or "None".
For any requirement with "None" or "Partial" coverage:
- If it logically belongs to an existing plan, update that plan and add the
  requirement to its source-references section
- Otherwise, create a new plan file with the next available number
STEP 2 — DESIGN DECISION AUDIT
Review every "Key design decisions" section across all plan files. Verify:
- Every decision is marked RESOLVED (not OPEN)
```

```
- No two plans resolve the same question with contradictory answers
- Every decision references its source requirement or constraint
If any decisions are OPEN:
- Propose a resolution with rationale
- Flag it prominently for human review
Produce a summary table in 00-coverage-matrix.md under a "## Design Decisions"
heading:
| # | Decision | Plan File | Status | Notes |
|---|----------|-----------|--------|-------|
STEP 3 — INTER-PLAN CONSISTENCY CHECK
For every pair of plans that share a dependency or touch the same component:
- Verify they don't make conflicting assumptions about shared interfaces
  (API contracts, database schema, shared types)
- Verify dependency directions are consistent (if Plan A says it depends on
  Plan B, Plan B must not also depend on Plan A — no circular dependencies)
- Verify no two plans claim ownership of creating the same artifact
  (table, endpoint, class) without one explicitly depending on the other
Document any conflicts found and resolve them by updating the affected plans.
STEP 4 — CROSS-CUTTING CONCERNS CHECK
Verify that the following concerns are addressed in at least one plan
(or documented as not applicable with justification):
- Authentication and authorization
- Data protection and privacy (applicable regulations)
- Accessibility (applicable standards)
- Observability (logging, metrics, health checks)
- Deployment and infrastructure
- Error handling and resilience
- Backward compatibility / migration path- Performance requirements (if specified in
tendering)
For any concern not covered, either:
- Add it to the most relevant existing plan, or
- Create a cross-cutting plan file (e.g., "XX-cross-cutting-concerns.md")
STEP 5 — HANDBOOK COMPLIANCE CHECK
Review the plans against each applicable handbook section. Verify:
- Technology choices match the approved technology stack
- Architecture patterns follow the prescribed structure
- No plan introduces a technology, pattern, or approach that contradicts
  the handbooks
- Any necessary exceptions are documented with justification
STEP 6 — SUMMARY AND SIGN-OFF READINESS
Produce a summary at the top of 00-coverage-matrix.md with:
**Coverage Statistics:**
- Total SHALL/SKAL requirements: [N]
- Fully covered: [N] ([%])
- Partially covered: [N] ([%])
- Not covered: [N] ([%])
```

```
**Design Decision Status:**
- Total decisions: [N]
- Resolved: [N]
- Open (requires human input): [N]
**Consistency Issues Found:** [N] — [resolved/pending]
**Cross-Cutting Gaps Found:** [N] — [resolved/pending]
**Handbook Conflicts Found:** [N] — [resolved/pending]
**Ready for Phase 2:** Yes / No — [reason if No]
ITERATION:
If you made changes to any plan files during this review, re-run Steps 1–5
on the updated plans. Repeat until no new gaps are found.
If any requirement is ambiguous or appears to conflict with another
requirement, document the conflict with your proposed resolution and flag it
for human review. Do not silently skip ambiguous requirements.
```

23.4 Áfangi 2: Framkvæmdaáætlun (Execution Plan)

Markmið áfangans er að umbreyta áætlunarskjölum Áfanga 1 í **vinnulotu áætlun** þar sem hver vinnulota er sjálfstæð vinnueining sem gervigreindar kóðunaragent getur lokið í einni keyrslu.

Mikilvægt: Kjarni þessa áfanga er að framleiða *afritunarhæfar vinnulotu kvaðningar* — fullbúnar kvaðningar sem útfærsluaðili getur afritað beint inn í kóðunaragent (t.d. Devin.AI) án nokkurra breytinga. Framkvæmdaáætlun án afritunarhæfra vinnulotu kvaðninga uppfyllir **ekki** kröfur viðaukans.

23.4.1 Lágmarksinnihald framkvæmdaáætlunar

Hluti	**Lýsing**
Heildaryfirlit	Lýsing á tilgangi áætlunarinnar og tengslum við áætlunarskjöl
Framkvæmdargraf (Mermaid)	Myndræn framsetning á háðum milli vinnulota
Vinnulotu listi	Töluleg upptalning á vinnulotum, raðað eftir keyrsluröð
MVP merkingar	Skýr merking á hvaða session mynda fyrstu keyrsluhæfu MVP útfærsluna
Forgangsröðun	Í hvaða röð á að keyra vinnulotur, með rökstuðningi
Vinnulotu kvaðningar	**Afritunarhæfar kvaðningar** í code-blokkum, einn fyrir hverja vinnulotu
Sannvottunarviðmið	Hver vinnulota skal innihalda gæða viðmið til að meta hvort verkinu sé lokið

23.4.2 Skrásafnsgerð fyrir framkvæmdaáætlun

Framkvæmdaáætlanir skulu vistaðar í /docs/execution/:

```
-docs/
└── docs/
    ├── plans/
    │   └── ...
    └── execution/
        ├── 00-master-sequence.md          # Heildaröð allra session
        ├── 00-dependency-graph.md         # Mermaid-myndi yfir öll háð
        ├── 01-patient-onboarding-execution.md
        ├── 02-clinician-workflows-execution.md
        └── ...
```

Heiti skráa parast við áætlunarskjölin — 01-patient-onboarding.md í /plans/ á sér samsvarandi 01-patient-onboarding-execution.md í /execution/.

23.4.3 Stærð vinnulota

Atriði	Viðmið
Æskileg stærð á session	200–800 línur af kóðabreytingum (prófunarkóði og skjölun undanskilin)
Hámarkstími á vinnulotu (AI agent)	Ein keyrsluvinnsla, engin mannleg inngrip nauðsynleg innan vinnulotu
Fjöldi skráa í vinnulotu	Hægt að halda utan um í einni breytingabeiðni (PR) fyrir hverja kóðageymslu þar sem hver PR hefur skýrt afmarkað svið.
Vinnulota = ein PR per kóðageymslu	Hver vinnulota skal samsvara einni breytingabeiðni í Pull Request fyrirkomulagi (kafli 4.6) fyrir hverja kóðageymslu. Vinnulotur geta innihaldið margar PR fyrir mismunandi hluta verkefnisins (sbr. framenda og bakenda breytingar)

23.4.4 Uppbygging kvaðninga fyrir hverja vinnulotu

Hver afritunarhæf vinnulotu kvaðning skal innihalda eftirfarandi þætti:

Þáttur	Tilgangur
Samhengi	Hvar í lausninni vinnulotan passar inn
.agent/ lestur	Skylda til að lesa viðeigandi staðla úr .agent/standards/ áður en hafist er handa
Tilvísanir í hönnun	Tenglar í Figma frame eða skjámyndaheiti
Aðgerðir	Nákvæm lýsing á hvaða skrár á að breyta eða búa til
Viðtökuviðmið	Mælanleg viðmið um lokið verk (build passes, tests pass, lint passes, manual checks)
Branch- og commit-snið	Heiti kóðagreinar og commit-skilaboða (samkvæmt kafla 1.6)
Skil	Hvaða samantekt agent á að skila í lokin

23.4.5 Dæmi um kvaðningar (Phase 2 prompts)

Execution-plan generation prompt:

```
You are creating a session-based execution plan that AI coding agents
will run to implement the features described in the implementation plans.

The quality of this execution plan determines whether AI agents can execute
autonomously. Every session prompt must be self-contained and copy-pasteable
into an AI coding agent session with zero modifications.

INPUT:
- Read ALL implementation plan documents in /docs/plans/ thoroughly
- Read ALL existing code in the target repositories to understand current
  patterns, conventions, file locations, and naming
- Read .agent/standards/ for coding standards

OUTPUT:
For each implementation plan file in /docs/plans/, produce a corresponding
```

```
execution plan file in /docs/execution/ with matching numbering:
  /docs/plans/01-feature-area.md → /docs/execution/01-feature-area-execution.md

Also produce:
  /docs/execution/00-master-sequence.md — single ordered list of every session
    across all execution plans, in recommended execution order
  /docs/execution/00-dependency-graph.md — full Mermaid graph of all
    inter-session dependencies across the project

EXECUTION PLAN DOCUMENT STRUCTURE:
Each execution plan file must follow this exact structure:

### 1. Title
  # <NN> — <Feature Area> Execution Plan

### 2. Overview (max 10 lines)
  What this plan covers, which components it affects, and key design
  decisions (all resolved). Bullet the critical resolved decisions so agents
  see them immediately.

### 3. Reference Documents
  Table with columns: Ref | Location/Path | Role.
  The implementation plan is always the "Primary reference". Include section
  numbers (§) that each session will need.

### 4. Target Repositories
  Table with columns: Repository | URL | Changes summary.
  List every repo that will receive PRs from this plan.

### 5. Critical Operational Constraints
  Hard rules that apply to ALL sessions in this plan. Common examples:
  - Session size limit (e.g., "Each session should target 200–800 lines
    of code changes")
  - Format preservation rules
  - Security invariants (e.g., "never call X from frontend")
  - Data isolation rules
  - Character encoding requirements
  - Branch and PR naming conventions

### 6. Secrets Inventory
  Table with columns: Secret | Used By | Already Exists.
  State explicitly if no new secrets are required.

### 7. Resolved Decisions (if the plan had design questions)
  Table with columns: # | Question | Decision | Impl Plan Ref.
  All questions must be resolved. This table is authoritative for all
```

```
  sessions.

### 8. Dependency Graph
  ASCII art diagram showing session dependencies with arrows.
  Below the diagram, write out:
  - Execution order (which sessions must be sequential, which can parallel)
  - WHY certain sessions cannot run in parallel (e.g., "both modify the
    same file")
  - Recommended sequential order for simplicity

### 9. Session Details
  For EACH session, use this exact sub-structure:

  #### Session S<N> — <Descriptive Name>

  **Implementation plan refs:** §X.Y, §X.Z (specific section numbers)
  **Repo target:** `Org/repo-name`
  **Depends on:** S<N-1> (what must be complete), or "None (first session)"
  **Estimated complexity:** S / M / L (with line-count range, e.g., ~400 lines)
  **New secrets:** None (or list them)

  ##### What This Session Delivers

  Numbered list of EVERY deliverable. Group by category using **bold
  headers** (e.g., "**SQL migrations:**", "**TypeScript types:**",
  "**Unit tests:**"). For each item:
  - Specify the EXACT file path to create or modify
  - Specify exact class names, method signatures, column definitions
  - Include code blocks with exact DDL, DTOs, or interface definitions
    when the implementation plan provides them
  - Reference specific lines/patterns in existing code to follow
    (e.g., "follow the pattern at TokenManager.cs:232-317")
  - Mark CRITICAL items that agents must not deviate from

  ##### Verification
  Bullet list of checks that must pass before the session is complete:
  - Specific lint/build/test commands
  - Specific behavioural checks
  - "CI passes on PR" (always last)

  End with: "Do NOT proceed to Session S<N+1>. Report results and stop."

### 10. Session Budget Summary
  Table with columns: Session | Repo(s) | Est. Size | Primary Deliverables.
  "Est. Size" is S / M / L with approximate line count (e.g., "M (~500 lines)").
  Include a Total row.
```

```
### 11. Cross-Cutting Technical Notes
  - **What NOT to Duplicate** — existing code to reuse, not reinvent
  - **Reusable Components** — table of components with Location and Purpose
  - **Invariants** — rules that apply to every session (e.g., "every SQL
    query on TableX MUST include WHERE clause Y")
  - **Naming Conventions** — namespace prefixes, branch prefixes, etc.

### 12. Do's and Don'ts for AI Agents Executing This Plan
  Two bullet lists:
  **DO:** — positive instructions with specific code/pattern references
  **DON'T:** — explicit prohibitions with reasons

### 13. Session Prompts
  Intro line: "Copy-paste the exact prompt below into a new AI coding
  agent session to start each session. Each prompt is self-contained."

  For EACH session, a fenced code block containing the copy-paste prompt.
  Each prompt must follow this internal structure:

  ```
  <One-line imperative summary of what to implement>. Execute "Session S<N>
  — <Name>" from the execution plan at "<path-to-this-file>" in the
  <repo-name> repo. The full technical design is in "<path-to-impl-plan>"
  (especially §X.Y for <topic>, §X.Z for <topic>). Repo: Org/repo-name.
  The repo uses `<default-branch>`.

  <Prior session status — which sessions are COMPLETE, what artifacts exist
  as a result. Be specific about what tables/classes/endpoints now exist.>

  Implement in <repo-name> ONLY.

  <ALL-CAPS SECTION HEADER FOR DELIVERABLE GROUP 1>:
  1. <Exact deliverable with file path, class name, columns, etc.>
  2. <Next deliverable...>

  <ALL-CAPS SECTION HEADER FOR DELIVERABLE GROUP 2>:
  3. <Exact deliverable...>

  ...continue for all deliverable groups...

  VERIFICATION:
  - <specific check>
  - <specific check>
  - CI passes on PR.
```
```

```
  Do NOT proceed to Session S<N+1>. Report results and stop.
  ```

14. Post-Implementation
  What to do after all sessions merge:
  - Monitoring plan (what to watch, what alerts to set)
  - Validation after real data accumulates
  - Future enhancements explicitly marked as out of scope

15. References
  Bullet list of all referenced documents, repos, and patterns.

SESSION SIZING GUIDELINES:
- Target 200–800 lines of code changes per session (tests and docs excluded)
- Each session = one PR per repo
- A session may touch multiple repos only if the changes are tightly coupled
  and small
- If a deliverable is too large for one session, split it. Prefer splitting
  by layer (DB → API → UI) rather than by feature slice
- A session must be completable by an AI coding agent in a single run
  without human intervention

SESSION PROMPT QUALITY CRITERIA:
A well-written session prompt must pass this test: "An AI coding agent that
has never seen this project should be able to execute the prompt as-is and
produce a correct PR without asking any clarifying questions."

To achieve this, each prompt must:
- Name every file to create or modify (full repo path)
- Name every class, method, table, and column
- Reference specific existing code patterns by file:line
- Include exact DDL, DTOs, or signatures from the implementation plan
- State what prior sessions delivered (so the agent knows what exists)
- State which repo to target and which branch convention to follow
- Include a verification checklist the agent can run mechanically
- End with a stop guard preventing premature continuation

HARD CONSTRAINTS:
- All documents in English
- All design decisions must be RESOLVED before sessions can be written
- Session prompts must be copy-pasteable with zero modification
- Every session must have a verification section
- Every session must end with "Do NOT proceed" and "Report results and stop"
- Branch naming must follow the project's convention (e.g., feature/ prefix)
- The dependency graph must be correct — no session may reference artifacts
  that a prior session has not yet created.
```
```

23.4.6 Dæmi um afritunarhæfa kvaðningu

Eftirfarandi er dæmi um kvaðningu fyrir vinnulotu. Útfærsluaðili notar kvaðninguna óbreytta í kóðunaragent.

```
SESSION S03 — Patient onboarding API endpoint
Plan: 01-patient-onboarding.md
Branch: feature/patient-onboarding-s03
Estimated size: ~450 lines
Depends on: S01 (DTOs and shared contracts), S02 (Dapper repository skeleton)

Read these standards first (do not skip):
- .agent/standards/api/rest-conventions.md
- .agent/standards/api/validation.md
- .agent/standards/database/dapper.md
- .agent/standards/dotnet/code-style.md

Design references:
- Figma frame: Patient-Onboarding-Form (frame ID 142:3870)
- Screen inventory entry: SCR-PT-01

Session S01 and S02 are COMPLETE. The following now exist:
- PatientOnboardingRequest DTO at src/Patients.Api/Contracts/
- PatientRepository skeleton at src/Patients.Infrastructure/Repositories/
- Database migration 002 with the Patients table

Implement in the-project-api ONLY.

ENDPOINT:
1. Create POST /api/patients/onboard endpoint in
   src/Patients.Api/Endpoints/Patients/OnboardPatientEndpoint.cs
   following the FastEndpoints REPR pattern used in the project.
2. Route: POST /api/patients/onboard
3. Auth: API key with scope "write:patients:onboard"
4. Request DTO: PatientOnboardingRequest (created in S01). Fields:
   NationalId (string, required, 10 chars), FullName (string, required),
   DateOfBirth (DateOnly, required), ContactPhone (string, optional).
5. Response: PatientOnboardingResponse with PatientId (long).

VALIDATION:
6. Create src/Patients.Api/Validators/OnboardPatientValidator.cs
   using FluentValidation:
   - NationalId: not empty, exactly 10 characters, digits only
   - FullName: not empty, max 200 characters
   - DateOfBirth: not in the future
7. Return 400 ProblemDetails on validation failure.

REPOSITORY:
8. Implement PatientRepository.CreateAsync in
```

```
   src/Patients.Infrastructure/Repositories/PatientRepository.cs:
   - INSERT into Patients table using Dapper
   - Return the new SCOPE_IDENTITY()
   - If duplicate kennitala, throw DuplicatePatientException
9. Catch DuplicatePatientException in the endpoint → return 409.

UNIT TESTS:
10. Tests in Tests/UnitTests/Endpoints/OnboardPatientEndpointTests.cs:
    - Valid request → 201 with PatientId
    - Invalid NationalId (too short) → 400
    - Missing FullName → 400
    - Future DateOfBirth → 400
    - Duplicate kennitala → 409
    - All inputs validated before reaching repository
11. Tests achieve ≥80% coverage on new files.

VERIFICATION:
- Endpoint responds at POST /api/patients/onboard
- Returns 201 with patient ID on success
- Returns 400 with ProblemDetails on invalid input
- Returns 409 if kennitala already exists
- No raw input reaches the repository
- All existing tests still pass
- dotnet build succeeds with zero warnings
- dotnet test passes
- Lint/format clean
- CI passes on PR

Do NOT proceed to Session S04. Report results and stop.
```

23.5 Áfangi 3: Skil á framkvæmdaráætlun og rýni

Þegar áfangar 1 og 2 eru kláraðir skal útfærsluaðili skila inn áætlunar- og framkvæmdaskjöl í -docs kóðageymslu verkefnisins og óska eftir rýni frá tæknilegum ábyrgðaraðila verkkaupa (SH).

23.5.1 Skil í kóðageymslu

Krafa	**Lýsing**
Geymsla	Allar áætlanir og framkvæmdaáætlanir vistaðar í -docs kóðageymslu verkefnisins
Útgáfustýring	Skil í gegnum breytingabeiðni (Pull Request) eins og annar kóði og skjölun (kafli 4.6)
Heitið á breytingabeiðni	feature/planning-NN eða feature/execution-NN eftir áfanga
Engar áætlanir utan kóðageymslu	Áætlanir í einkaskýjum verksala (t.d. Notion, Confluence) teljast ekki til skila

23.5.2 Rýni og samþykki verkkaupa

Tæknilegur ábyrgðaraðili SH framkvæmir rýni á áætlunum áður en útfærsla session prompta hefst.

Rýniviðmið SH	Spurning
Þekja á SKAL-kröfum	Er sérhver SKAL-krafa útboðs og þróunarhandbókar þakin í áætlun og framkvæmdaáætlun?
Tæknileg samræmi	Eru áætlanir í samræmi við Tæknistefnu og kafla 8 í þróunarhandbók?
Hönnunarsamræmi	Vísa kvaðningar í rétt Figma frame og skjámyndaheiti?
Framkvæmd	Er framkvæmdaráætlun raunhæf og vel skilgreind?
MVP afmörkun	Mynda MVP-merktar vinnulotur keyrsluhæfa virkni (end-to-end slice)?
Gæði kvaðninga	Eru kvaðningar afritunarhæfar — þ.e. þurfa þær engar viðbótarútskýringar?
Samhæfni við handbók	Er ekkert í andstöðu við Þ-viðauka, Þróunarhandbók, Hönnunarhandbók eða Rekstrarhandbók?

23.5.3 Gæðahlið fyrir útfærslu

Krafa	Tegund
Útfærsla session prompta skal **ekki** hefjast fyrr en rýni er lokið og samþykki SH liggur fyrir	SKAL
Niðurstaða rýnis er skráð sem PR comment og skjalfest í /docs/execution/00-approval-log.md	SKAL
Frávik frá samþykktri framkvæmdaáætlun á útfærslustigi krefjast nýrrar rýni eða formlegrar breytingar (sjá kafla 1.6 og viðauka „Yfirferðarskjal kóðaryni")	SKAL

23.6 Áfangi 4: Útfærsla vinnulotu kvaðninga

Þegar rýni er lokið og samþykki SH liggur fyrir hefst útfærsla. Útfærsluaðili keyrir kvaðningar í þeirri röð sem framkvæmdaáætlunin skilgreinir.

23.6.1 Keyrslurröð

Atriði	Krafa
Sjálfgefin röð	Master-sequence í /docs/execution/00-master-sequence.md
Samhliða keyrsla	Heimil ef áætlun leyfir og útfærsluaðili getur tryggt að engar tvær vinnulotur breyti sömu skrám
Frávik frá röð	Skjalfest í /docs/execution/00-execution-log.md með rökstuðningi
MVP áfangi	Allar MVP-merktar vinnulotur skulu kláraðar í einum samfelldum áfanga áður en aðrar lotur hefjast, nema annað sé samþykkt af SH

23.6.2 Sannvottun á útfærslu hverrar vinnulotu

Útfærsluaðili ber ábyrgð á að sannvotta útfærslu kóðunaragenta í lok hverrar vinnulotu. Gervigreindin er smíðatæki — *ekki staðgengill mannlegrar fagþekkingar* (sjá kafla 11).

Sannvottunarstig	Hver framkvæmir	Hvað er sannvottað
Sjálfvirkt	CI píplína (sjá kafla 6)	Bygging, prófanir, kóðagreining, gæðahlið
Kóðaryni	Forritari hjá útfærsluaðila	Réttmæti, öryggi, samræmi við hönnun, samræmi við .agent/ staðla
Virknisprófanir	Prófari/forritari hjá útfærsluaðila	Virkni samkvæmt viðtökuviðmiðum vinnulotu og kvaðningar
Hönnun	Hönnuður eða forritari með hönnunarþekkingu hjá útfærsluaðila	Samræmi við Figma frame og hönnunarstaðla

Verkkaupi	SH (skv. kafla 5 — viðtökuferill)	Formleg móttöku yfirferð á breytingabeiðni

Engar vinnulotur teljast kláraðar fyrr en öll viðeigandi sannvottunarstig hafa verið framkvæmd og skráð.

23.6.3 Viðhald .agent/ möppu á meðan á útfærslu stendur

.agent/ mappa skal uppfærast samhliða útfærslu eins og skilgreint er í kafla 11:

- Hönnunarákvarðanir sem teknar eru á smíðastigi skulu skjalfestar í .agent/specs/ eða viðeigandi standards-skrá.
- Ef vinnulota leiðir í ljós vöntun á staðlaskrá skal sú skrá vera bætt við í sömu eða næstu breytingabeiðni.
- Lokakröfur Leiðar B (kafli 5.2.2 og 11) gilda óbreyttar.

23.7 Áfangi 5: Lokastaðfesting og prófanir

Eftir að allar vinnulotur úr framkvæmdaáætluninni eru kláraðar skal útfærsluaðili framkvæma lokastaðfestingu á heildarlausninni áður en formleg afhending fer fram til verkkaupa.

23.7.1 Lokaprófanir

Prófunartegund	**Krafa**
End-to-end virknisprófanir	Yfir alla MVP- og post-MVP virkni samkvæmt áætlunarskjölum
Prófanir á samþættingum	Allar ytri samþættingar (FHIR, IdP, ytri kerfi) prófaðar í prófunarumhverfi (PPT)
Öryggisprófanir	Skv. kafla 12 — auðkenning, aðgangsstýring, OWASP Top 10
Aðgengisprófanir	Skv. kafla 14 — WCAG 2.2 AA + valdar AAA viðmið
Þekjuprófanir	≥80% á nýjum kóða og ≥65% heildarþekja (kafli 13)
Lækningatækjamat	Endurmat ef lýsing breyttist á meðan á þróun stóð (kafli 3.4)
EU AI Act mat	Endurmat ef gervigreindarvirkni í afurð breyttist (kafli 8B og Tæknistefna §10)

23.7.2 Skjölun og afhending

Krafa	**Lýsing**
Skjölun lausnar	Uppfærð samkvæmt viðauka „Skipulag og snið hugbúnaðarskjölunar"
Útfærsluskýrsla	/docs/execution/99-completion-report.md með yfirliti yfir kláraðar vinnulotur, frávik og óklárað eða opin álitaefni.
.agent/ mappa	Endurspegli endanlega stöðu lausnar (kafli 11)
Yfirferðarskjal kóðarýni	Skv. viðauka „Yfirferðarskjal kóðarýni"
Gátlistar	Gátlistar 1–6 í viðauka „Gátlistar við verkefnastýringu" útfylltir

23.7.3 Gæðahlið fyrir afhendingu

Engin formleg afhending hefst fyrr en:

1. Allar MVP vinnulotur eru kláraðar og sannvottaðar.

2. Allar samþykktar post-MVP vinnulotur eru kláraðar og sannvottaðar — eða skjalfest sem útfellt verk/hætt við með samþykki SH.
3. Lokaskjölun og útfærsluskýrsla eru í -docs kóðageymslu.
4. Verkkaupi hefur framkvæmt formlega kóðarýni og samþykkt skv. kafla 5.

23.8 Skjalaskipulag í -docs kóðageymslu

```
-docs/
└── docs/
    ├── plans/                          # Áfangi 1
    │   ├── 00-overview.md
    │   ├── 00-coverage-matrix.md
    │   ├── 01-<feature-area>.md
    │   ├── 02-<feature-area>.md
    │   └── ...
    ├── execution/                       # Áfangi 2
    │   ├── 00-master-sequence.md
    │   ├── 00-dependency-graph.md
    │   ├── 00-approval-log.md           # Áfangi 3
    │   ├── 00-execution-log.md          # Áfangi 4
    │   ├── 99-completion-report.md      # Áfangi 5
    │   ├── 01-<feature-area>-execution.md
    │   ├── 02-<feature-area>-execution.md
    │   └── ...
    ├── architecture.md
    ├── deployment.md
    └── ...
```

Ofangreint skipulag bætist við þá /docs möppuskipan sem skilgreind er í viðauka „Skipulag og snið hugbúnaðarskjölunar".

23.9 Kröfur til útfærsluaðila – samantekt

Flokkur	Krafa	Tegund
Áætlanagerð	Allar áætlunar- og framkvæmdaskrár vistaðar í -docs kóðageymslu og númeraðar frá 01	SKAL
Áætlanagerð	Áætlun og framkvæmdaáætlun skrifaðar í markdown sniði	SKAL
Áætlanagerð	Notkun langhliða gervigreindarvélar fyrir áfanga 1 og 2 (t.d. Devin.AI eða sambærilegt)	GETUR
Áætlanagerð	Ítrun við gervigreindarvél og sjálfsmat á þekju áður en framkvæmdaáætlun er sett í rýni	SKAL
Framkvæmdaáætlun	Framkvæmdaáætlun skal innihalda afritunarhæfar kvaðningar	SKAL
Framkvæmdaáætlun	Háðsgraf í Mermaid-sniði í hverri framkvæmdaáætlun	SKAL
Framkvæmdaáætlun	MVP-merking á þeim vinnulotum sem mynda fyrstu keyrsluhæfu þverskurðarvirkni	SKAL
Gap-rýni	Engin útfærsla session prompta hefst fyrr en gap-rýni SH er lokið og samþykki liggur fyrir	SKAL
Gap-rýni	Niðurstaða gap-rýni skjalfest í /docs/execution/00-approval-log.md	SKAL
Útfærsla	Mannleg sannvottun á útfærslu hverrar session áður en næsta vinnulota hefst	SKAL

Útfærsla	Frávik frá samþykktri framkvæmdaáætlun krefjast nýs rýnis eða formlegrar breytingar	SKAL
Útfærsla	.agent/ mappa uppfærist samhliða útfærslu, ekki einungis í lok verkefnis	SKAL
Lokaprófanir	Lokaprófanir og útfærsluskýrsla kláruð áður en formleg afhending hefst	SKAL
Almennt	Verksali ber fulla ábyrgð á öllum kóða, óháð uppruna (sjá kafla 11)	SKAL
Almennt	Verksali skal upplýsa verkkaupa um öll gervigreindarverkfæri sem notuð eru í ferlinu (sjá kafla 11)	SKAL

24. Viðauki – Hugbúnaður sem lækningatæki (SaMD/MDSW)

Þessi viðauki lýsir regluverki, stöðlum og ferli sem gilda þegar hugbúnaður sem verkkaupi þróar eða pantar telst lækningatæki samkvæmt lögum nr. 132/2020 og reglugerð (ESB) 2017/745 (MDR).

Viðaukinn er ætlaður sem viðmiðunarskjal — hann kemur ekki í stað faglegs ráðgjafa á sviði lækningatækja og regluverks. Ef hugbúnaður er flokkaður sem lækningatæki er mælt með að leita til sérfræðinga í regluverki lækningatækja (regulatory affairs).

24.1 Hvenær er hugbúnaður lækningatæki?

Flokkunarmat byggist á tilgangi hugbúnaðarins. Ákvarðanaferlið er útfært sem tafla í meginmáli þróunarhandbókar (sjá undirkafla „Hugbúnaður sem lækningatæki — flokkun og ferli"). Opinber leiðbeining um matið er MDCG 2019-11 Rev.1 sem skilgreinir ákvarðanatré og dæmi.

Ef hugbúnaður telst lækningatæki fer flokkun í áhættuflokk (I, IIa, IIb, III) eftir **Rule 11** í Viðauka VIII við MDR. Í stuttu máli: hugbúnaður sem veitir upplýsingar til greiningar- eða meðferðarákvörðunar er að lágmarki í flokki IIa, og hækkar eftir alvarleika mögulegra afleiðinga. Nákvæm flokkunarregla er í MDR Viðauka VIII, reglu 11.

24.2 Dæmi úr starfsemi verkkaupa

Eftirfarandi dæmi eru til leiðbeiningar. Endanleg flokkun fer ávallt eftir raunverulegri tilgangslýsingu.

Hugbúnaður	**Lækningatæki?**	**Líklegur flokkur**	**Skýring**
Sjúkraskrárkerfi sem geymir og birtir klínísk gögn	Nei	—	Einungis geymsla og birting, engin úrvinnsla í læknisfræðilegum tilgangi
Verkbeiðnakerfi (order entry) sem sendir fyrirmæli milli kerfa	Nei	—	Samskipti og flutningur gagna, engin sjálfstæð greining
Klínískt ákvörðunarstuðningskerfi sem greinir gögn og leggur til meðferð	Já	IIa	Upplýsingar til meðferðarákvörðunar (Rule 11a)
Myndgreiningarhugbúnaður sem aukar birtingu DICOM-mynda til greiningar	Já	IIa	Úrvinnsla á gögnum í greiningartilgangi (Rule 11a)
Lyfjavíxlverkanakerfi sem varar við hættulegum lyfjasamsetnimgum	Já	IIa–IIb	Upplýsingar til meðferðarákvörðunar; flokkur háður alvarleika (Rule 11a)
Gervigreindarkerfi sem spáir fyrir um sepsis út frá lífsmörkum	Já	IIb–III	Spá um lífshættulegt ástand (Rule 11a/b)
Gervigreindarkerfi sem greinir húðbreytingar á myndum	Já	IIa–IIb	Greining sjúkdóms (Rule 11a)
Tímasetningarkerfi starfsmanna	Nei	—	Enginn læknisfræðilegur tilgangur
Fjareftirlit (remote monitoring) sem vaktar blóðþrýsting sjúklinga	Já	IIa–IIb	Vöktun lífsmarka (Rule 11b)

24.3 Viðeigandi staðlar

Staðall	Svið	Samhæfður undir MDR?
ISO 13485:2016	Gæðastjórnunarkerfi framleiðanda	Já
IEC 62304:2006+A1:2015	Hugbúnaðarlíftími og öryggisflokkun	Já
ISO 14971:2019	Áhættustýring	Já
IEC 62366-1:2015	Notkunarfræði (usability engineering)	Já
IEC 82304-1:2016	Öryggi sjálfstæðs heilsuhugbúnaðar	Nei (í vaxandi notkun)

24.4 Samræmismat og CE-merking

Hugbúnaður sem telst lækningatæki skal vera CE-merktur áður en hann er tekinn í notkun. Samræmismatsleið fer eftir áhættuflokki: flokkur I leyfir innra mat (self-certification), en **flokkur IIa og hærri krefst aðkomu tilkynnts aðila** (Notified Body) sem úttektar gæðakerfi og tækniskjölun.

Enginn tilkynntur aðili starfar á Íslandi; framleiðendur þurfa að leita til tilkynnts aðila í öðru EES-ríki. Listi yfir tilkynnta aðila er á NANDO-gagnagrunninum (sjá heimildatöflu).

Nánari upplýsingar um samræmismatsaðferðir eru í MDR gr. 52 og Viðaukum IX–XI.

24.5 EUDAMED – skráningarskylda

Framleiðendum er skylt að skrá sig og lækningatæki sín í EUDAMED (European Database on Medical Devices). Nákvæmar dagsetningar og leiðbeiningar eru á eudamed.eu. Verkkaupi skal fylgjast með gildandi tímafrestum og tryggja skráningu áður en lækningatæki er sett á markað.

24.6 Tengsl við kröfur handbókakerfisins

Margar kröfur þróunarhandbókarinnar skarast við kröfur IEC 62304 og ISO 14971. Eftirfarandi tafla sýnir hvar skörunin er mest og hvar viðbótarkröfur koma til.

Kafli handbókar	Sambærileg IEC 62304 krafa	Munur / viðbót
Kafli 4 (Kóðageymslur, breytingastýring)	§8 Configuration management	IEC 62304 krefst formlegrar útgáfustjórnunar á öllum hugbúnaðareiningum, ekki eingöngu kóða
Kafli 5 (Viðtöku- og útgáfuferlar)	§5.8 Software integration, §5.7 Verification	IEC 62304 krefst skjalfestrar rekjanleika frá kröfum til prófana (traceability matrix)
Kafli 6 (Gæðaferlar)	§5.1 Software development planning	IEC 62304 krefst formlegrar þróunaráætlunar með öryggisflokkun
Kafli 13 (Prófanir)	§5.5 Detailed design verification, §5.6–5.7	IEC 62304 kvarðar prófanir eftir öryggisflokki; flokkur C krefst einingaprófana á öllu
Kafli 15 (Hugbúnaðarskjölun)	§5.1–5.8, MDR Viðauki II	MDR krefst tækniskjölunar í ákveðnu sniði (Technical Documentation) sem nær langt umfram /Docs möppu
Kafli 19 (Rekstur og eftirlit)	§6 Software maintenance, §7 Risk management	IEC 62304 krefst formlegrar viðhaldsáætlunar og áhrifamats á allar breytingar

24.7 Lykilheimildir

Heimild	Slóð / tilvísun
Lög nr. 132/2020	althingi.is/lagas/nuna/2020132.html
Reglugerð (ESB) 2017/745 (MDR)	eur-lex.europa.eu
MDCG 2019-11 Rev.1 (flokkun hugbúnaðar)	health.ec.europa.eu
MDCG 2020-1 (klínískt mat hugbúnaðar)	health.ec.europa.eu
MDCG 2025-6 (AI Act og MDR samspil)	health.ec.europa.eu
Lyfjastofnun — lækningatæki	lyfjastofnun.is/laekningataeki
Lyfjastofnun — skráning dreifingaraðila	lyfjastofnun.is/laekningataeki/skraning-framleidanda
NANDO gagnagrunnur (tilkynntir aðilar)	ec.europa.eu/growth/tools-databases/nando
EUDAMED	ec.europa.eu/tools/eudamed