HÁSKÓLI ÍSLANDS
Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild

HBV403G: Vefforritun 2 · Vor 2025

Andri Fannar Kristjánsson

Kröfuumsjónarkerfi fyrir Hugbúnaðarverkefni · Skýrsla

---

1 Inngangur

Kröfugreining er einstaklega mikilvæg er kemur að hugbúnaðargerð. Kröfugreining tryggir að hugbúnaðurinn sé framleiddur á réttan hátt, með færri villum og í samræmi við allar kröfur sem viðskiptavinir hafa sett fram. Því er mikilvægt að hafa góðan hugbúnað sem sér um að halda utan um alla eiginleika og kröfur sem verkefni kunna að hafa. Þetta varð mér ljóst eftir að hafa tekið áfangann HBV301 Verkfræði Kröfugreiningar, þar sem farið var yfir kröfugreiningu og mikilvægi hennar við nákvæma hugbúnaðargerð. Í bókinni Software Requirements eftir Karl Wiegers og Joy Beatty, þar sem vel er farið yfir alla eiginleika kröfugreiningar, kemur það bersýnilega í ljós að það að geyma kröfur á reglubundinn hátt í sérstökum hugbúnaði myndi einfalda til muna allt bókhald, breytingar, og vitnanir sem er notast mikið við. Því kom það vel til greina sem einstaklingsverkefni í Vefforritun 2.

2 Útfærsla

Þar sem fullbúinn hugbúnaður með það hlutverk að halda utan um allar kröfur sem kunna að koma til við hugbúnaðargerð er sér í lagi stór þegar allt er tekið með, mun þetta einstaklingsverkefni aðeins hafa það að markmiði að gera kleift að búa til Notkunardæmi (*e. Use Cases*) á góðan máta, með öllum þeim eiginleikum sem því fylgja. Eftirfarandi virkni er til staðar sem uppfyllir það sem sett var fram í verkefnalýsingunni, og einnig kröfur Hópverkefnis 1 (sjá einnig kafla 3 fyrir nánari upplýsingar um hvaða tækni var notuð). Notendur geta þannig:

- Búið til aðgang og skráð sig inn
- Búið sér til verkefni fyrir hvern hugbúnað sem notandi vill safna kröfum um. Fyrir hvert verkefni er svo hægt að:
  - Hlaða upp einni mynd af UML klasariti eða öðru tengdu verkefninu
  - Skilgreina og halda utan um:
    - **Leikara** (*e. Actors*), sem hafa Nafn og Lýsingu
    - **Viðskiptareglur** (*e. Business Rules*), sem hafa Skilgreiningu, Tegund, Breytanleika (*e. Mutability*) og Uppruna
    - **Notkunardæmi** (*e. Use Cases*), sem hafa:
      - Einfalda reiti, þ.e. Nafn, Höfundur, Dagsetning, Lýsing, Orsakavaldur, Forgangsröðun, Tíðni, Aðrar Upplýsingar og Ályktanir
      - **Aðal- og Aukaleikara** (*e. Primary & Secondary Actors*), ýmist leikarar sem eru til í verkefninu eða nýir
      - **For- og Eftirskilyrði** (*e. Pre- & Postconditions*), sem geyma aðeins Lýsingu

- **Venjulegt, Önnur og Undantekningarflæði** (*e. Normal, Alternate & Exception Flows*), sem innihalda skref sem leikari getur tekið þegar rennt er í gegn um Notkunardæmið
- **Viðskiptareglur** (*e. Business Rules*), ýmist reglur sem eru til í verkefninu eða nýjar

Eftirfarandi eiginleikar eiga svo við:

- Hægt verður að vitna í Viðskiptareglur, Skilyrði, Flæði og Skref í flæðum þegar skref eru skilgreind, og þannig hægt að hoppa á milli staða.
- Þegar upplýsingum er breytt er gamla útgáfan geymd, og hægt verður að skoða og afturkalla breytingar (*í vinnslu*).
- Notendur með Admin réttindi geta sótt allar upplýsingar geymdar í gagnagrunni, þ.e. öll verkefni, öll notkunardæmi o.þ.h.
- Allar upplýsingar frá notendum eru hreinsaðar.
- UML klasarit fyrir einindi (*e. Entities*) var sett upp og uppfært eftir þörfum. Sjá á GitHub síðu verkefnisins.
- Framendi verkefnisins er aðgengilegur hér (*í vinnslu*).

Þessi virkni var fyrir valinu þar sem talið er að hann yrði mest notaður, hefur ýmsa mismunandi þætti sem krefjast sniðugra lausna, og er mikilvægur hluti af Vision & Scope skjali. Hugsað er að seinna meir væri hægt að halda áfram og bæta við frekari virkni, með það markmið að geyma allar þær upplýsingar sem þarf til að búa til Vision & Scope og SRS skjöl.

3 Tækni

Verkefnið notar Node.js og TypeScript. Bakendinn notar Hono til að útfæra REST-ful API, og Zod til að passa upp á týpur. Gögn eru vistuð í PostgreSQL gagnagrunn með Prisma og Cloudinary til að geyma myndir. Framendinn notar Next.js og Tailwind CSS.

4 Hvað Gekk Vel

Í heild gekk verkefnið vel, þ.e.a.s. það var ekkert sem þurfti alfarið að hætta við, þó nokkrir hnökrar hafi komið upp. Gott var að koma hönnuninni niður á blað í upphafi með því að setja upp UML-rit og uppfæra það eftir þörfum með útfærslunni. Bakendavinnan með Hono gekk einnig fremur vel, og var auðvelt að tileinkna sér.

5 Hvað Gekk Illa

Ekkert í verkefninu gekk sérstaklega illa, en margir hlutir tóku langan tíma að útfæra á góðan hátt, og fór því mikill tími í ýmsa hluti. Í flestum öðrum verkefnum í Háskóla Íslands hefur verið notast við hlutbundin forritunarmál á borð við Java, og því er nokkuð góð reynsla í að nota slík mál, og áætlað var að geta nýtt sérstaklega reynsluna á að hafa þróað REST-ful API með Java Spring, en erfitt reyndist að nota mál eins og JavaScript (TypeScript gerði oft hlutina nokkuð skárri en þó ekki mikið) sem heldur ekki utan um týpur á sama hátt (*e. loosely typed*). Mikið af tíma fór því í að passa upp á að flóknar týpur skiluðu sér rétt í og úr gagnagrunni, og skipta þeim upp eftir því hvaða gögn ættu og hver mættu vera til staðar

hverju sinni. Þá þurfti að vinna öðruvísi með tengingar á milli hluta, sem virka vel þegar notað er t.d. Java Spring, en þarfnast öðruvísi útfærslu hér. Mikill tími fór einnig í að passa upp á hreinsun gagna og að notendur geta aðeins átt við eigin hluti.

6 Hvað Var Áhugavert

Margt var áhugavert við verkefnið, og var mikið lært af því að gera slíkt verkefni á þessum skala. Í rauninni má segja að þetta verkefni er álíka stórt og verkefni sem fjögurra einstaklinga teymi gera í Hugbúnaðarverkefni 1. Allt sem var notað í þessu verkefni er fremur nýtt og lítil reynsla er í að nota slíka tækni, svo þetta var gríðarlega gott tækifæri til að læra. Einnig var skemmtilegt að finna út úr öllum tengingunum og eiginleikunum sem þurftu að vera til staðar í einindunum svo hægt væri að tengja alla hluti vel saman.

7 Samantekt

Verkefnið gekk vel, og náðist að gera flest sem ákveðið var. Það tók töluvert lengri tíma að framkvæma en ætlast var í upphafi. Marga hluti væri sennilega hægt að gera betur, en að mestu leyti er verkefnið á góðum stað. Það var unnið á fagmannlegan hátt, með ítarlegri skjölun, snyrtilegum kóða og með öryggi í huga. Ef það hefði verið forritað með tækni sem reynsla var af að nota, eins og Java Spring, hefði það vissulega gengið hraðar fyrir sig, en stór hluti af verkefninu var að læra á nýja tækni. Gott hefði verið ef tímanum hefði verið varið á skilvirkari hátt, en þegar ný tækni er notuð, þótt hún sé fljótlærð, gerast hlutirnir hægar.
Í framhaldi er sýnin sú að gera hugbúnaðinn líkum GitHub nema fyrir kröfugreiningu, þ.e. hægt verður að bjóða fleirum í að gera breytingar, verkefni geta verið sýnileg eða ósýnileg öðrum o.þ.h.. Svo mætti einnig bæta við stuðning fyrir fleiri upplýsingum um verkefni á borð við hluthafa (*e. Stakeholders*), notendakröfur (*e. User Requirements*), gæðaeiginleika (*e. Quality Attributes*), virknikröfur (*e. Functional Requirements*) o.s.frv.