Voorbereiding bachelorproef

Dit document is de eerste stap richting jouw bachelorproef. Vul alle gegevens hieronder in. Dien nadien als pdf-document op Leho in (cursus Bachelorproef, onderdeel Opdrachten). Bezorg het document ook per e-mail aan jouw interne promotor en vraag hem/haar naar feedback.

Deadline: vrijdag 18 februari 2022 - 12u.

1.1 Algemeen

Jouw voornaam & naam:	Branco Bruyneel
Jouw coach van researchproject:	Dieter De Preester
Jouw interne promotor:	Wouter Gevaert

1.2 Contractplan

Wat was jouw onderzoeksvraag?

Hoe bouw je een Full Stack speed typing test applicatie in Rust & WebAssembly?

Wat was het vooropgestelde technische onderzoek?

Om de huidige status van Full Stack web development in Rust te onderzoeken zal ik een webapplicatie maken van kop tot teen. Alles zal dan uiteraard worden geschreven in Rust van frontend tot backend.

De web app zal een typetest applicatie zijn waar een gebruiker zijn typesnelheid zal kunnen testen, oefenen & bestuderen. De gebruiker zal kunnen inloggen via SSO en zo zijn afgelegde testen kunnen bijhouden. Hij kan de resultaten bekijken met behulp van visualisaties op zijn profiel pagina. Doordat ik de resultaten zal opslaan zal er een database & API aan te pas komen.

Wat waren de vooropgestelde succescriteria?

De taal onder de knie hebben met een werkende demo die die volgende criteria bevat:

- De statische resultaten worden bewaard in een database
- De frontend kan de resultaten ophalen via een API uit de database
- De applicatie is beveiligd met SSO
- Er is een type test pagina aanwezig
- Er is een profiel pagina aanwezig

1.3 Zelfreflectie over researchproject

Welke succescriteria zijn (niet) behaald?

Door dat het programmeren van het project moeizaam verliep heb ik de succescriteria wat versimpeld. Zo krijgt de gebruiker op de webpagina een random code snippet te zien die hij zo snel mogelijk moet overtypen. Daarna worden zijn resultaten getoond op een simpele resultaten pagina. Daar kan hij zijn tijd, woorden per minuut, nauwkeurigheid en fouten zien. De resultaten worden niet bewaard sinds er geen tijd was voor een login systeem.

De code snippets worden per programmeertaal bewaard in een SQLite database. Met een simpele API kunnen er CRUD acties worden uitgevoerd op de database. De frontend spreekt de API aan voor een random snippet op te halen.

Wat verliep vlot?

Het compileren van Rust naar WebAssembly ging vrij moeiteloos. Wat ook vlot ging was het opzetten van TailwindCSS voor dit project.

Wat kon beter?

De manier hoe de code snippets nu worden bewaard in de database is eerst de tabs & newlines om te zetten naar “\t” & “\n”. Hier heb ik vlug een parser tool voorgeschreven. De reden hiervoor is dat de frontend de code snippets geformatteerd moet kunnen ophalen. Deze manier van werken is redelijk omslachtig en kan verbeterd worden.

In de API heb ik gebruik ik schema types die ik normaal kan hergebruiken in de frontend sinds de frontend & api in een workspace zitten. Maar als ik de schema types wou importeren in de frontend kreeg ik compilatie errors dat sommige backend packages niet konden gecompileerd worden naar WebAssembly. Omdat ik weinig tijd had heb ik dit vlug opgelost door de schema types te herdefinieerd in de frontend.

Wat was de feedback van de jury?

/

1.4 Reflectie met externen

Inhoudelijk

- Hoe men Rust naar productie deployed
- Betere aanpak om de code snippets geformatteerd door te sturen tussen de api & frontend

Met wie zal je reflecteren over researchproject?

In een discord community van de frontend library yew heb ik een core maintainer leren kennen die werkt bij een bedrijf waar heel hun stack met Rust geschreven is. Het zou leuk zijn mocht hij de tijd hebben om met mij over een paar zaken te reflecteren.