Rakendusest

Tegemist on hambaarsti juurde registreerimise rakendusega, nagu lähteülesandes kirjas. Kasutajal on võimalik registreerida visiite, mille sisenditeks on kolm välja, arst, visiidi algus ja visiidi lõpp.

Arst valitakse etteantud nimekirjast. Juhul kui sisestatakse arst, keda ei ole olemas (näiteks muudetakse brauseris HTML elementi), näidatakse kasutajale vastavat veateadet.

Visiidi algus valitakse *timepicker*-st, kus kasutaja valib graafiliselt kalendrilt kuupäeva ja määrab kellaaja. Visiidi lõpp valitakse samamoodi, aga kasutaja saab graafiliselt valida ainult kellaaega, sest hambaarsti visiidid minuteada üle mitme päeva ei kesta.

Timepicker teeb sisestatud kuupäevast ja kellaajast lahtritesse korrektse formaadiga sõned, lisaks ei ole võimalik sisestada mineviku kuupäevasid. Visiidi varasem alguskellaaeg on kell 08:00 ja hiliseim lõppkellaaeg on 18:00, selline on näidishambaarstide tööpäev. Visiidi lõpp peab olema pärast visiidi algust.

Kui kõik sisendid on olemas, saab kasutaja kontrollida sisestatud ajavahemiku kattuvust sisestatud arsti ülejäänud visiitidega, ehk visiidid saavad kattuda ajaliselt, kui nendel visiitidel on erinevad arstid. Kui kuskil sisestatud andmetes on viga, antakse sellest kasutajale lahtrite juurde veateatega teada. Kui kõik on korras ja aeg ei kattu, antakse sellest kasutajale teada.

Teine valik, mida kasutaja teha saab, on visiidi registreerimine. Kui kuskil on viga, antakse samamoodi kasutajale sellest teada, kui registreerimine õnnestus, viiakse kasutaja lehele kus sellest teada antakse ja kus kasutajal on võimalus uuele registreerimisele suunduda või vaadata kõikide visiitide nimekirja.

Kõikide visiitide nimekirja lehel saab kasutaja filtreerida tabelit arstide järgi. Kui üritatakse filtreerida arsti järgi, keda ei eksisteeri, näidatakse veateadet. Vaikimisi kuvatakse kõikide arstide visiite. Visiidile peale vajutades saab kasutaja siseneda detailvaatesse, kus on võimalik visiit ära kustutada või selle andmeid muuta. Andmete muutmisel kehtivad samasugused reeglid nagu registreerimisel, ehk arst eksisteerib süsteemis, aeg ei ole minevikus, visiit ei kattu arsti teiste visiitidega.

Igal lehel on võimalik muuta rakendust eesti- või inglisekeelseks.

Võimalikud modifikatsioonid

Praegu on andmebaasis ainult visiidi isendid, kus arst on määratud sõnena. Kui oleks rakendus suurem ja peaks toetama erinevaid operatsioone ka arstidega, võiks teha eraldi arsti isendid andmebaasi ning need visiitide isenditega ühendada. Praeguse funktsionaalsusega ei ole see minu arvates väga vajalik.

Registreerimisele annaks juurde lisada erinevaid asjakohaseid lahtreid/sisendeid visiitide kohta.

Lisatud teegid

Lisaks olemasolevatele raamistikele ja teekidele, on rakendusse lisatud veel:

- Flatpickr (https://flatpickr.js.org) *Timepicker* visiidi alguse ja lõpu jaoks.
- Bootstrap (https://getbootstrap.com) Rakenduse kujundus
- jQuery (https://jquery.com) Bootstrapi jaoks on seda vaja
- Popper.js (https://popper.js.org) Bootstrapi jaoks on seda vaja

Mis oli raske?

Java olen varem kasutanud, aga Springi sai esimest korda nähtud. Põhiline raskus oligi aru saada, mida kõike Spring teeb ning kuidas seda kasutada. Alguses sai näiteks uuritud, mida erinevad annotatsioonid teevad, milliseid on vaja kasutada ja kuidas neid kasutada. Töö käigus sai ka avastatud, et Springi kasutades on üsna oluline isendite getter- ja settermeetodid ning nende nimetus erinevate annotatsioonidega.

Rakenduse visuaalse osa juures puutusin esmakordselt kokku Thymeleafiga. Ka selle erinevaid võimalusi ja kasutamisi sai alguses mõnda aega uuritud.

DTO ja DAO lühendid olid alguses natukene võõrad, kuid internetist leidis nende kohta piisavalt informatsiooni.

Mis oli lihtne?

MVC arhitektuuriga olen varem kokku puutunud, seega kui tundmatud ja võõrad Springi lisandid veidikene tuttavamaks said, oli projekti ülesehitus arusaadav ja kerge ise funktsionaalsust juurde lisada.

Thymeleafiga oli kerge dünaamilisust juurde lisada, kui sellega natuke tutvutud sai. Sama lugu Springiga sisendite valideerimine.

Ajakulu

Umbkaudselt jagunes kogu rakenduse tegemise ajakulu järgmiselt:

- 40% Springi ja Thymeleafi avastamine ning erinevate võimaluste uurimine, näiteks otsingud stiilis „spring check if date is in the past”.
- 40% leitud võimaluste rakendamine ja töölesaamine ning üldine koodi kirjutamine.
- 10% Projekti seadistamine, koodi javadoci lisamine, kommentaaride lisamine, jUnit testid, koodi refactorimine.