TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Rasmus Talioja

Veebipõhine koondpaneel Tartu rattaringluse andmete visualiseerimiseks ja analüüsimiseks

Bakalaureusetöö (9 EAP)

Juhendaja:
Helen Tera, MSc

Tartu 2025

Veebipõhine koondpaneel Tartu rattaringluse andmete visualiseerimiseks ja analüüsimiseks

Lühikokkuvõte:

Tartu rattaringlus on viie aasta jooksul kogunud suure hulga sõidu- ja GPS-andmeid, kuid nende analüüsimiseks puudus terviklik interaktiivne tööriist. Käesoleva lõputöö eesmärk oli arendada veebipõhine koondpaneel nende andmete visualiseerimiseks ja analüüsimiseks. Rakendus loodi Svelte/SvelteKiti, MapLibre ja DuckDB abil. Peamine väljakutse oli suuremahuliste ruumiliste andmestike visualiseerimine. See lahendati andmete töötlemisega vektorpaanideks kasutades PMTiles formaati. Tulemuseks on funktsionaalne koondpaneel, mis võimaldab rattaringluse kasutuse ajaloolist ja ruumilist analüüsi, kuvades võtmenäitajaid, jaamapõhiseid andmeid ning teekondade ja kuumkaartide visualiseerimisi kaardil.

Võtmesõnad: Andmete visualiseerimine, veebirakendus, koondpaneel, rattaringlus, liikuvusandmed

CERCS: P175 Informaatika, süsteemiteooria

Interactive Web Dashboard for Tartu Smart Bike Data Visualization and Analysis

Abstract:

Tartu bike-sharing has collected a large volume of ride and GPS data over five years but lacking a comprehensive interactive analysis tool. The objective of this thesis was to develop a web-based dashboard for visualizing and analyzing this data. The application was built using Svelte/SvelteKit, MapLibre and DuckDB. The main challenge was visualizing large spatial datasets. This was solved by processing data into vector tiles using PMTiles format. The result is a functional dashboard enabling temporal and spatial analysis of bike-sharing usage, including key metrics, station-based data and map visualizations of routes and heatmaps.

Keywords: Data visualization, webapp, dashboard, bike sharing, mobility data

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

Tartu rattaringlus on jalgrattalaenutuse süsteem Tartus, mis pakub linlastele taskukohast ja keskkonnasäästlikku liikumisvõimalust. Rattaringluse tegevusaja jooksul on kogunenud märkimisväärne hulk täpseid sõidu- ning GPS-andmed.

Selline andmestik võimaldab teadlastel ja poliitiliste valikute kujundajatel teha paremaid otsuseid jalgrattasõbraliku linnaruumi planeerimisel. Kuigi osaliselt on neid andmeid kasutatud ka varasemates uurimisprojektides, puudus terviklik tööriist, mis võimaldaks andmestiku ülevaatlikku käsitlemist kogu süsteemi senise tööperioodi lõikes.

Jalgrattaringluste andmeid analüüsivatest ja visualiseerivatest veebirakendustest on avalikult kättesaadavad vaid vähesed. Järgnevalt tutvustab autor neist mõnda. Selline analüüs võimaldab tutvuda käesoleva lõputöö käigus loodud rakendusele sarnanevate lahendustega ning annab ülevaate nende funktsionaalsustest.

Google Looker Studio platvormil loodud Capital Bikeshare DC jalgrattaringluse koondpaneel [1] pakub kasutajatele võimalust interaktiivselt uurida sõitude arvusid läbi erinevate filtrite kuupäeva, kellaaja või nädalapäeva lõikes. Otsingut saab kitsendada jaama, kuupäeva, rattatüübi ja liikmelisuse järgi. Diagrammide juures olev kaart visualiseerib jaamade kasutatavust ning geograafilisi asukohti. Looker Studio eeliseks on kasutajasõbralik liides, kuid see võib jääda piiravaks keerukamate analüütiliste vajaduste korral.

Capital Bikeshare DC andmestikul põhineb ka teine, GoodGuyThor rattaringluse koondpaneel [2]. Paneel on loodud Vega-Altair[1] Pythoni andmevisualiseerimise teeki kasutades. See keskendub jalgrattaringluse andmete seostamisele mitmesuguste teguritega nagu ilmastikuolud ja nädalapäevad. Rakenduse tugevuseks on lihtne ja kiire arendusprotsess, kuid visualiseerimiste paindlikkus ning suuremate andmemahtude käsitlemise võimekus saaksid olla paremad.

Lisaks avalikele veebirakendustele on analüüsitud valminud rakendusi ka teadustöödes. Üheks selliseks on Cortez, Sánchez ja Vázquezi artikkel [3]. Autorid keskenduvad visuaalse tööriista arendamisele, mis võimaldab analüüsida väiksemate jalgrattaringluse süsteemide kasutustrende. Tööriist esitab Logroño linna BiciLog rattaringluse andmeid erineva ajalisuse

[1] https://altair-viz.github.io

detailsusega[2], kuvab liiklusvooge jaamade vahel ning pakub põhjalikku ülevaadet kasutajate andmetest, sealhulgas registreeritud elukoha, läbitud vahemaa ja soopõhiste mustrite alusel. Cortez, Sánchez ja Vázquezi loodud rakendus on väljatoodud rakendustest kõige laiaotstarbelisem ning võimaldab artikli põhjal rohkelt erinevaid uurimissuundi.

Käesoleva lõputöö eesmärk on arendada veebipõhine koondpaneel Tartu rattaringluse andmete visualiseerimiseks ja uurimiseks. Valminud koondpaneel võimaldab kasutajatel analüüsida rattaringluse kasutusmustreid nii ajalisest kui ka ruumilisest vaatepunktist lähtuvalt ning võrrelda erinevaid ajaperioode. Lõputöö tulemuseks on kasutajasõbralik veebirakendus, mis hõlbustab Tartu rattaringluse andmete interaktiivset uurimist ja mõtestamist.

Lõputöö jaguneb neljaks sisupeatükiks. Esimeses peatükis tutvustatakse koondpaneeli funktsionaalseid ja mittefunktsionaalseid nõudeid. Teises peatükis kirjeldab autor rakenduse arendusel kasutatud tehnoloogiaid. Kolmandas peatükis antakse ülevaade arhitektuurist ja arendusprotsessist. Viimases sisupeatükis analüüsib autor valminud rakendust ning pakub välja võimalikke edasiarendusi.

[2] Aasta, kuu, nädal või aastaaeg

2. Rakenduse nõuded

Selles peatükis tuuakse välja Tartu rattaringluse koondpaneeli rakenduse funktsionaalsed ja mittefunktsionaalseid nõuded. Lõputöö autori ülesanne oli kujundada rakenduse visuaalne pool, puhastada algandmed ning arendada rakendus kasutades sobivaid tehnoloogiaid.

Tartu rattaringluse andmeid uurib ja kasutab erinevates teadustöödes Tartu Ülikooli ITS (Intelligent Transportation Systems) labor, kes täitis rakenduse nõuete koostamisel tellija rolli. Seetõttu on ka rakenduse esialgseks sihtrühmaks ITS labor ja teised Tartu Ülikooli töötajad.

2.1 Funktsionaalsed nõuded

Selleks, et veebirakendus täidaks eesmärki ning oleks sihtrühmale kasulik, koostati funktsionaalsed nõuded. Nendeks on:

- Andmete filtreerimine ja agregeerimine erinevate ajaskaalade lõikes.
- Kogu olemasoleva andmestiku kasutamine aastatest 2019-2023.
- Võtmenäitajate kuvamine: sõitude koguarv, läbitud distants ja kogukestus.
- Jaamade nimekirja kuvamine.
- Interaktiivse Tartu linna kaardi kuvamine.
- Jaamade asukohtade kuvamine kaardil.
- Jaama valimise võimalus kaardilt.
- Jaama valimise võimalus jaamade nimekirjast.
- Jaamapõhiste näitajate kuvamine.
- GPS andmete kuvamine kaardil sobitatud teekondadena.
- GPS andmete kuvamine kaardil kuumkaardina (ingl *heatmap*).
- Filtreeritud andmete põhjal diagrammide kuvamine.
- Jaamapõhiste andmete põhjal diagrammide kuvamine.

2.2 Mittefunktsionaalsed nõuded

Selleks, et rakenduse kasutamine, edasiarendamine ja parandamine oleks mugav koostati mittefunktsionaalsed nõuded. Nõueteks on, et:

- Rakendus töötab veebikeskkonnas.
- Rakendus toetab veebilehitsejate Google Chrome, Mozilla Firefox, Microsoft Edge ja Safari kõige uuemaid versioone.
- Rakenduse lähtekood on kommenteeritud ja struktuurselt üles ehitatud nii, et võimaldab edasist arendamist.
- Juhtpaneeli jõudlus on mõistlik. Rakendus reageerib kasutaja tegevustele ning visualiseeringud ja filtrid uuenevad optimaalsel kiirusel, ilma et see mõjutaks kasutatavust või põhjustaks ressursitarbimise hüppelist kasvu.
- Rakendus mugandub erinevatele ekraanisuurustele.

3. Kasutatud tehnoloogiad

Käesolevas peatükis kirjeldab autor olulisemaid kasutatud tehnoloogiaid. Valikute tegemisel võeti arvesse tehnoloogiate sobivust nõuetega, nende kaasaegsust ning kasutusmugavust. Lisaks arvestas autor oma isiklikku kogemust ning huvi vastavate tehnoloogiate vastu.

3.1 Svelte

Järgnev peatükk on kirjutatud Svelte dokumentatsiooni põhjal [4].

Svelte on veebipõhiste kasutajaliideste loomiseks mõeldud raamistik. See kasutab kompilaatorit, et teisendada HTML-i, CSS-i ja JavaScripti abil kirjutatud deklaratiivsed komponendid minimaalselt koodi sisaldavaks ja hästi optimeeritud JavaScriptiks.

Svelte kõige olulisem funktsionaalsus on võimalus jagada aplikatsioon taaskasutatavateks komponentideks. Komponent ühendab kolm osa: struktuuri[3], funktsionaalsuse (JavaScript) ja stiili (CSS). Svelte lisab HTML-i laiendusi, mida kasutatakse reaktiivse sisu defineerimiseks. Seetõttu muutub JavaScripti oleku muutudes ka automaatselt tulemus. HTML-i osade kordamiseks või tingimisi kuvamiseks kasutatakse plokke, näiteks tingimus-, kordus- või ootusplokke. Samuti saab HTML siltide sisse lisada reaktiivseid avaldisi.

Reaktiivsed andmed defineeritakse `<script>` sildi sees. See kood käivitatakse komponendi eksemplari loomisel. Tipptasemel (ingl *top-level*) deklareeritud või imporditud muutujad on kasutatavad komponendi struktuuri osas.

Lisaks tavapärasele JavaScriptile võimaldab Svelte kasutada ruune (ingl *runes*), mille abil saab deklareerida komponendi sisendeid (ingl *props*) ning lisada komponenti reaktiivsust. Ruunid on sümbolid, mida kasutatakse ".svelte" failides Svelte kompilaatori käitumise juhtimiseks. Kui käsitleda Svelte'i kui keelt, siis on ruunid kui võtmesõnad (ingl *keywords*) ja osa selle süntaksist. Ruunid algavad sümboliga "$" ja näevad välja nagu funktsioonikutsed, kuid erinevalt funktsioonidest ei pea neid importima ega saa kasutada muutujate või argumentide väärtustena.

Svelte võimaldab kasutada TypeScripti JavaScripti asemel. TypeScripti kasutamiseks tuleb `<script>` sildil kasutada viidet `lang="ts"`. Sarnast viidet on võimalik kasutada ka `<style>` sildil, et võimaldada SCSS või PostCSS kasutamist. Svelte kompileerib mittestandardsed

[3] Svelte poolt laiendatud HTML

keeled automaatselt standardseteks[4]. Stiiliplokid on Sveltes vaikimisi lokaalsed (ingl *scoped*) ehk piiratud ulatusega, mis tähendab, et stiilid ei laiene komponendivälistele elementidele.

Svelte alternatiivid on React[5], Vue.js[6] ja Angular[7], kuid nende kasutamise mugavus on väiksem. Tarkvaraarendajate seas läbiviidud küsitluste [5] [6] kohaselt on Svelte kasutajad oma valikuga kõige rohkem rahul. 73% Sveltega töötanud arendajatest soovivad seda ka edaspidi kasutada. Autoril on varasem kogemus Vue.js ja Sveltega. Mainitud raamistikest peab autor aga just Svelte kasutamise mugavust parimaks

3.2 SvelteKit

Järgnev peatükk on kirjutatud Svelte dokumentatsiooni põhjal [4].

SvelteKit on raamistik, mis võimaldab kiiresti arendada töökindlaid ja suure jõudlusega veebirakendusi, kasutades Svelte'i kasutajaliideste arenduseks. Svelte kuvab kasutajaliidese komponente aidates luua ühe vaate, kuid SvelteKit hõlmab mitmesuguseid terviklikuks veebirakenduseks vajalikke funktsionaalsusi, näiteks:

- ruuterit, mis võimaldab kasutajaliidese dünaamilist uuendamist hüperlinkide kaudu navigeerimisel;
- kompileeritud koodi optimeerimist, mille eesmärk on laadida ainult rakenduse toimimiseks hädavajalik osa;
- võrguühenduseta töö toetust;
- lehtede eellaadimist (ingl *prefetch*) enne kasutaja navigeerimist;
- kuvamisprotsessi konfigureeritavust, võimaldades rakenduse osade renderdamist serveripoolselt (ingl *server-side rendering*, edaspidi SSR), kliendipoolselt (ingl *client-side rendering*, edaspidi CSR) või kompileerimise etapis vaated valmis ehitada (ingl *prerendering*);
- pildihalduse optimeerimist;

[4] HTML, CSS, JavaScript
[5] https://react.dev/
[6] https://vuejs.org/
[7] https://angular.dev/

SvelteKit kasutab failisüsteemi põhist marsruutimist ehk rakenduse URL-struktuur määratakse otse projekti kataloogipuu põhjal. See tähendab, et arendaja ei pea ruute käsitsi defineerima, vaid iga fail ja kaust kataloogis "src/routes" vastab konkreetsele marsruudile (ingl *route*) rakenduses. Iga marsruudi kataloog sisaldab ühte või mitut marsruudifaili, mis on tuvastatavad oma nime alguses oleva sümboli "+" järgi.

3.3 MapLibre

MapLibre GL JS on TypeScripti teek, mis võimaldab interaktiivsete kaartide renderdamist brauseris, kasutades WebGL-tehnoloogiat ning vektorpaanide (ingl vector tiles) andmeformaati [7]. WebGL on platvormist sõltumatu, litsentsitasuta avatud veebistandard, mis määratleb madala taseme 3D-graafika liidese (API), tuginedes OpenGL ES-ile. Liides on kättesaadav ECMAScriptile[8] HTML5 "canvas" elemendi kaudu [8].

Svelte MapLibre on arendusjärgus teek, mis pakub idiomaatilist Svelte'i tuge MapLibre GL kaardirakenduse integreerimiseks. Teek järgib Svelte arenduspõhimõtteid, võimaldades loomulikku ja süntaktiliselt ühtset integratsiooni. [9]

Arendusperioodi vältel lisandus alternatiivne teek – Svelte MapLibre GL. Erinevalt eelmisest keskendub see just Svelte 5 funktsionaalsusega ühilduvusele [10]. Teegi autor toob ka välja, et lisaks muule sai ta inspiratsiooni ka eelnevalt mainitud teegi funktsionaalsusest.

Autor alustas rakenduse kirjutamist Svelte MapLibre teegiga ning kaalus üleminekut uuele, Svelte MapLibre GL teegile selle parema ühilduvuse tõttu. Kuna olemasolev lahendus vastas vajadustele ja puudusi ei ilmnenud, otsustati jätkata esialgse valikuga.

3.4 DuckDB

Järgnev peatükk on kirjutatud DuckDB dokumentatsiooni põhjal [11].

DuckDB on relatsiooniline[9] andmebaasihaldussüsteem (ingl *database management system* ehk DBMS), mis toetab struktureeritud päringukeelt SQL (ingl *Structured Query Language*).

DuckDB, tuntud ka kui OLAP (ingl *online analytical processing*, ehk analüütiline veebitöötlus) on loodud toetama analüütiliste päringute koormusi. Need on enamasti keerukad ja suhteliselt pika kestusega ning töötlevad märkimisväärset osa andmekogumist. Näiteks

[8] JavaScripti standardiseeritud spetsifikatsioon
[9] tabelipõhine

teostavad need tervete tabelite koondarvutusi või mitme suuremahulise tabeli liitmist. Andmete muudatused on samuti eeldatavalt suuremahulised, hõlmates mitme rea lisamist või tervete tabeli osade muutmist või lisamist korraga.

Selle töökoormuse tõhusaks toetamiseks on oluline vähendada iga väärtuse töötlemiseks kuluvat protsessori tsüklite hulka. DuckDB kasutab vektoriseeritud päringutöötlusmootorit, mis põhineb veerupõhisel andmestruktuuril. Ühe operatsiooni käigus töödeldakse korraga suurt hulka väärtusi ehk vektorit. See lähenemine vähendab märkimisväärselt liigsete ressursside kasutust, mis on iseloomulik traditsioonilistele ridade kaupa töötlevatele süsteemidele, nagu PostgreSQL, MySQL või SQLite. Vektoriseeritud töötlus parandab olulisel määral OLAP päringute jõudlust.

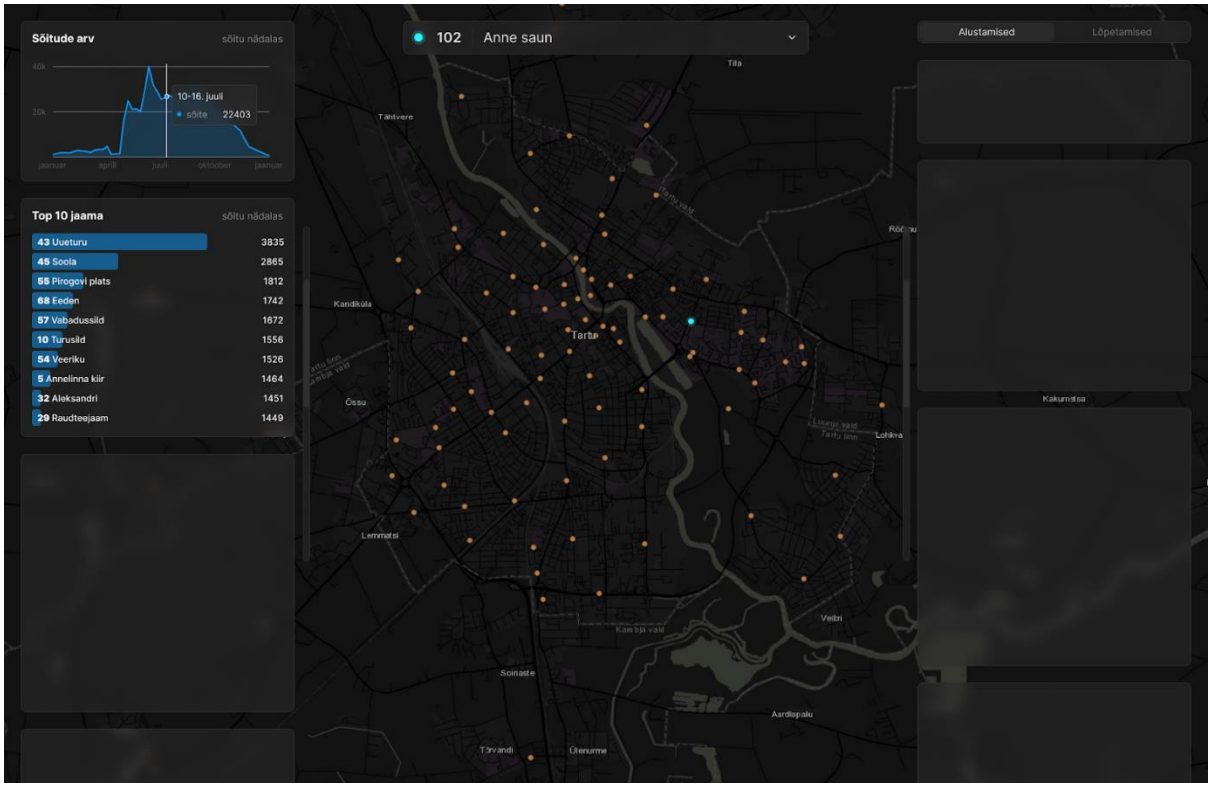
4. Rakenduse arhitektuur ja arendusprotsess

Selles peatükis annab töö autor ülevaate rakenduse arhitektuurist ja arendusprotsessist. Lisaks viidatakse kasutatud töövahenditele.

4.1 Rakenduse disain

Rakenduse prototüüp (Joonis 1) on disainitud kasutades digitoote disainimiseks loodud tarkvara Figma. Selle käigus katsetati erinevaid disainikeeli ning loodi kasutajakogemuse põhimõtted. Rakenduse disaini aluseks sai tervet ekraani kattev kaart ning ülejäänud osad disainiti arvestades sobivust sellega. Prototüüpi kasutas autor tutvustamaks ITS laborile oma visiooni ning hiljem arendusprotsessi käigus stiilijuhendina.

Rakenduse mõlemas ääres on külgpaneelid, millel on üksteise alla kuvatud diagramme ning võtmenäitajad. Kui diagramme sisaldav paneel ekraanile enam ei mahu, muutuvad külgpaneelid keritavaks. Ekraani ülemise ääre keskmises osas on kombineeritud sisend ehk liitboks (ingl *combobox*). Sisendi peale vajutades avaneb jaamade nimekiri ning filtrid. Samuti näitab see suletud olekus hetkel valitud jaama.



Joonis 1. Kuvatõmmis rakenduse prototüübi disainist.

Disainikeeleks valis autor klaasiefekti (ingl *glassmorphism*). IxDF väidab [12], et klaasiefekt on kaasaegne kasutajaliidese disainisuund, mida iseloomustab matistatud klaasi (ingl *frosted glass*) efekt, kus taust paistab läbi udustatud poolläbipaistvate paneelide. See stiil jäljendab klaasi ning loob sügavuse ja ruumilisuse tunde, säilitades samas modernse ja elegantse esteetika. Disainilahendus rõhutab visuaalset hierarhiat ja loetavust, võimaldades kasutajaliidesel tunduda kerge ja korrastatuna.

4.2 Andmete korrastamine ja andmebaasi kiht

Järgnevast peatükis tutvustab autor algandmeid ning nende puhastamist. Lisaks antakse ülevaade lõplikust andmebaasist.

4.2.1 Andmetega tutvumine

Autori esimeseks ülesandeks oli andmetega tutvumine ning nende korrastamine. Tartu rattaringluse andmetes oli palju puudujääke. Andmed olid CSV failides ning jaotatud kaheks: asukohtade failid ja sõitude failid. Asukohtade failid sisaldasid viie sekundi intervalliga GPS andmeid iga sõidu kohta. Sõitude failid sisaldasid iga sõidu kohta järgnevat infot: ratta number, kasutaja identifikaator (edaspidi id), ratta avamise aeg, ratta lukustamise aeg, sõidu alustamise jaam, sõidu alustamise jaama id, sõidu lõpetamise jaam, sõidu lõpetamise jaama id, pikkus kilomeetrites, kestus minutites, ratta tüüp[10], ning kasutaja sünniaasta.

Nii asukoha- kui ka sõidufaile oli iga kuu kohta üks või enam ning failid olid kohati erineva failinime struktuuriga. Lisaks olid need pakitud aastate kaupa kataloogidesse. Autor ühtlustas failinimed, et lihtsustada nende töötlemist. Järgnevalt valis autor OLAP koormustele mõeldud andmebaasi DuckDB. DuckDB toetab hulgi CSV failide lugemist kasutades metamärgi (ingl *glob*) süntaksit. DuckDB saab töötada nii püsirežiimis, kus andmed salvestatakse kettale, kui ka mälurežiimis, kus kogu andmestik hoitakse operatiivmälus [11]. Hulgi failide lugemine ning mälurežiimi kasutamine tegi andmete esialgse uurimise võimalikuks.

Algandmete veerud olid aja jooksul muutunud. Seetõttu lõi autor skripti, mis kaardistas andmete ajaloolist muutust. Veergude muutused on näha kuvatõmmisel (Joonis 2). Muutused on välja toodud järjestikku olevate failide erinevustega. Paljud veerud on lihtsalt nimekuju vahetanud, kuid siiski on lisatud ka uut ning eemaldatud vana infot. Selle info põhjal ühtlustas autor veerud programmaatiliselt.

[10] Tartu rattaringlusel on nii tava- kui ka elektrirattad

```
routes_2020_02.csv -> routes_2020_03.csv
Added columns:
        yearofbirth
Removed columns:
        dateofbirth

routes_2020_12.csv -> routes_2021_01.csv
Added columns:
        dateofbirth,
        PersonalIdCode
Removed columns:
        costs,
        yearofbirth,
        personalIdCode

routes_2021_06.csv -> routes_2021_07.csv
Removed columns:
        userID,
        rfidnumber,
        DurationMinutes,
        CycleType,
        Membership,
        dateofbirth,
        PersonalIdCode

routes_2021_07.csv -> routes_2021_08.csv
Added columns:
        userID,
        rfidnumber,
        DurationMinutes,
        CycleType,
        Membership,
        dateofbirth,
        PersonalIdCode
```

Joonis 2. Kuvatõmmis veergude muutustest.

Pärast veergude ühtlustamist luges autor andmed andmebaasi tabelitesse ning läks üle püsirežiimi kasutamisele. Lisaks lõi autor uue tabeli jaamade jaoks, kasutades Tartu linna rattaringluse avaandmeid[11].

4.2.2 Andmete puhastamine

Järgmiseks tööetapiks oli andmete puhastamine. Andmestiku sõitude failides oli iga sõidu alustamise ja lõpetamise jaama id sama, seetõttu pidi autor lähtuma algus- ja lõppjaamade nimedest. Jaamasid identifitseerivate numbrite ja nimede vahel leidus ebakõlasid. Numbrile 80 vastav jaam oli muutnud ainult nime kuid füüsiline jaam jäi samaks. Numbrile 70 vastav jaam

[11] https://geohub.tartulv.ee/datasets/Tartu::li-rattaringluse-parklad-avaandmed/about

oli viidud Rahingest üle Märjale. Tähelepanuväärne erisus oli numbrile 71 vastava jaama puhul, mis oli viidud Ujula tänavalt üle Ihastesse Pallase tänavale. Autor otsustas kasutada numbrile vastava jaama viimast nime.

Rattaringluse kasutuses on olnud ka ajutisi parklaid, näiteks suurürituste ajal, ning need on numbri asemel märgistatud lühenditega. Kuvatõmmisel on näited ajutistest parklatest ning nende esinemise sagedusest andmestikus (Joonis 3). Ajutistes parklates alustatud ja lõpetatud sõidud otsustas autor eemaldada andmete vähesuse tõttu. Samuti eemaldas autor puuduva algus- või lõppjaamaga andmed[12]. Eemaldatud andmed moodustasid kõigist andmetest 0,6%.

IP	14
Lauluväljak	2
MP	146
PM	5
Rahu8	2
SD	55
TDÖÖ	14
TK	1004
Undetermined	22816
WH	171

Joonis 3. Kuvatõmmis mittenumbrilistest identifikaatoritest.

Autor märkas, et andmestikust puuduvad 2023. aasta mai GPS-andmed, 2023. aasta mai viimase nädala sõitude andmed ning 2021. aasta juuni sõitude kasutajat või lisainfot puudutavad andmed (Joonis 2).

4.2.3 Lõplik andmebaas

Teekondade visualiseerimise tarbeks saatis käesoleva töö juhendaja 2020. aasta sõitude GPS-andmete kaardiga sobitamise (ingl *map-matching*) tulemused. Need teekonnad luges autor samuti andmebaasi sisse. Kaardiga sobitamine koostab ideaalsetel tingimustel teekonna realistliku trajektoori füüsiliste teede tegelike koordinaatide põhjal. Selline lähenemine töötab hästi mootorsõidukite teekondade kaardistamiseks, kuid kergliikurite teekonnad on ettearvamatud. Juhendaja kasutatas Valhalla[13] tarkvara, mis kasutab OpenStreetMap[14] andmestikku. Andmestikust on puudu mitmed kergliiklusteed ning muud kergliikuriga

[12] märgitud kuvatõmmisel „Undetermined“
[13] https://valhalla.github.io/valhalla/
[14] https://www.openstreetmap.org/about

läbitavad lõigud. GPS andmete ja sobitatud teekondade erinevust on näha ka rakenduses, kus mõlema andmestiku põhjal on olemas kaardipõhised visuaalid.

Puhastatud andmebaasis on info 3,87 miljoni sõidu kohta ning 385,8 miljonit rida GPS andmeid. Lisaks on andmebaasis 846,5 tuhande sõidu sobitatud teekonnad. Andmebaasifaili maht on 9,6 GiB. Eeltöötluses kasutatakse kõiki andmeid. Töötavas rakenduses kasutatakse ainult jaamade ja sõitude tabeleid. Optimeeritud andmebaasifaili suurus on 122 MiB.

4.3 Kasutajaliidese ja komponentide kiht

Järgnevas peatükis kirjeldab autor kasutajaliidese arendamise protsessi. Kirjeldatakse SvelteKiti projekti loomist, abivahendeid komponentide loomisel ning erinevaid lähenemisi andmete ruumilisele visualiseerimisele.

4.3.1 Projekti loomine ja seadistamine

Svelte projekti loomiseks soovitab ametlik dokumentatsioon käsku `pnpx sv create` Järgnevalt kirjeldab autor Sv tööriista kasutades Svelte dokumentatsiooni [4]. Sv on Svelte teegi arendajate meeskonna poolt loodud käsurea tööriist (ingl *command line interface*, ehk CLI). Tööriista funktsionaalsuste hulka kuuluvad mitmed käsud:

- `create` käsk loob uue SvelteKiti projekti. Selle argumentide hulka kuulub muuhulgas projekti malli valik[15], TypeScripti või JSDoci kasutamine[16], ning paljude ametlike lisade kaasamise valik;
- `add` käsk lisab SvelteKiti projekti uusi funktsionaalsusi, mille integratsioon raamistikuga on kontrollitud SvelteKiti arendusmeeskonna poolt. Valikus on näiteks: koodiformaadi ja -analüüsi teegid ESLint ja Prettier, kompilaatoripõhine i18n-teek[17] (ingl *internationalisation library*) Paraglide, testimisraamistikud Playwright, Storybook ja Vitest ning CSS-raamistik TailwindCSS[18];
- `check` käsk tuvastab projekti vigu ja hoiatusi, sealhulgas kasutamata CSS-i, ligipääsetavuse soovitusi (*accessibility hints*) ning TypeScripti kompilaatorivigu;

[15] valida saab näidis-, minimaalse- või teegiprojekti vahel
[16] Svelte soovitab kasutada tüübikontrolli
[17] lisab rakendusele mitme keele toetuse
[18] https://tailwindcss.com/

- `migrate` käsk automatiseerib koodibaasi migratsiooni Svelte ja SvelteKiti uutele versioonidele. Muudatused mis tuleb teha manuaalselt leiab üles käsu poolt lisatud `@migration` märgistustega;

Autor tegi sv CLI minimaalse TypeScripti malliga uue projekti. Lisamiskäsuga paigaldati ESLint, Prettier ning stiiliraamistik TailwindCSS.

4.3.2 Komponenditeek

Järgnevalt valis autor komponentide loomiseks teegi Melt UI [13], mille arendusele on autor ka ise kaasa aidanud. Erinevalt traditsioonilistest komponente pakkuvatest teekidest pakub Melt UI *buildereid*[19]. *Builderid* on funktsioonid, mis genereerivad omaduste ja atribuutide kogu, mida saab rakendada ükskõik millisele HTML-elemendile või Svelte komponendile. See võimaldab suuremat paindlikkust kasutajaliidese loomiseks.

Melt UI teeki kasutas autor nuppude, ajavahemiku sisendi ning kombineeritud sisendi funktsionaalsuste põhjana. Komponentide soovituslik ehitus ja HTML-elementide kasutus on välja toodud dokumentatsioonis. Teegi esimeses versioonis on dokumentatsioonis saadaval ka näidistel kasutatavad TailwindCSS-i klassid. Autor lisas komponentidele tavalise CSS-i stiili Figmas kujundatud disaini põhjal.

4.3.3 Andmete visualiseerimine

Autor uuris mitmeid andmete visualiseerimise teeke. Väljavalituks osutus komponentide teek LayerChart [14], millel oli kõige rohkem erinevaid funktsionaalsusi. LayerChart mähib enda komponentides diagrammiteegi LayerCake [15] komponente, kuid lisab nendele oma kujunduse ning mõned funktsionaalsused. LayerCake sobib ka iseseisvalt käesoleva projekti arendamiseks, kuid selle õpikõver on suurem. Aja kokkuhoiu eesmärgil otsustas autor kasutada LayerCharti poolt pakutavaid komponente mille kasutamine ning kohandamine on lihtsam.

LayerChart vajab töötamiseks TailwindCSS-i, mistõttu lisas autor selle projekti. Pärast TailwindCSS-i seadistamist sai autor alustada diagrammide loomist. Rakenduses kasutatakse lõputöö esitamise hetkel vertikaalseid ja horisontaalseid tulpdiagramme ning mitmekihilist radiaaldiagrammi.

[19] tuleneb samanimelisest programmeerimismustrist

4.3.4 Kaardipõhine visualiseerimine

MapLibre seadistamisel probleeme ei esinenud. Autor määras kaardi keskpunkti, suurendusastme ja valis kaardi aluskihi. Autor valis Carto[20] poolt tasuta kättesaadava aluskihi nimega „dark-matter“, mis oli visuaalselt sobivaim Figmas kujundatud disainikeelega. Järgmisena lisas autor kaardile andmebaasist päritud jaamade asukohamärgised.

Kõige aeganõudvam osa tööprotsessist oli andmete ruumiline visualiseerimine. Esialgses lahenduses teisendati andmebaasipäringute tulemused serveri poolel GeoJSON formaati ning tagastati kasutajale. Ühe jaama kohta leidub miljoneid ridu koordinaate ja muid GPS-andmeid. Seetõttu ulatusid päringute suurused sadade mebibaitideni. Sellega kaasnes ülekoormus brauserile, mis proovis kõiki andmeid korraga töödelda ning kuvada tulemust kaardil. Kokku võttis sellise lähenemisega ühe jaama päring aega vähemalt pool minutit ning populaarsemate jaamade puhul lõpetas brauser töö täielikult.

Selline lahendus ei vastanud mittefunktsionaalsetele nõuetele ning autor pidi proovima teist lähenemist. Kaardi aluskihi käitumisest inspireerituna leidis autor, et MapLibre toetab lisaks aluskaardile ka teisi vektorpaanide kihte. Uus lahendus oli GPS- ja sobitatud teekondade andmed eeltöödelda PMTiles [16] formaati. Selleks kasutas autor Tippecanoe [17] tööriista. Tippecanoe võimaldab töödelda joongeomeetriate puhul GeoJSON formaati ning punktgeomeetriate ehk koordinaatide puhul CSV faile. Samuti võimaldab see väiksematel suurendusastmetel lihtsustada geomeetriat, mida ei saa ekraanil eristada. Paanideks jagamine ja andmete lihtsustamine vähendavad päringute mahtu ning parandavad oluliselt rakenduse jõudlust.

Sobitatud teekondade vektorpaanideks töötlemiseks lõi autor skripti, kus iga jaama kohta loetakse andmebaasist vastavad koordinaadid ning teistendatakse need GeoJSON formaati. Järgmisena edastab skript käsurealt kõikide jaamade GeoJSON failid korraga Tippecanoe tööriistale, mis kompileerib ühe PMTiles arhiivi, kus iga faili info on leitav eraldi kihist.

GPS andmete vektorpaanideks töötlemisele lähenes autor sarnaselt, kuid andmete rohkuse tõttu pidi autor GeoJSONi asemel kasutama kompaktsemat CSV formaati, mida Tippecanoe punktgeomeetria puhul toetab. Skripti töö tulemusel loodi iga jaama kohta eraldi PMTiles arhiiv.

[20] https://basemaps.cartocdn.com

Faile hoitakse staatiliselt kompileeritud rakenduse kaustas. Rakendus kasutab HTTP *range*-päringuid[21], et pärida PMTiles ainult vajaduspõhiseid paane [18].

Rakenduses on võimalik näha kaardi pealmistes kihtides valitud jaamast alanud sõitude sobitatud teekondi ja kuumkaarti GPS-andmete koordinaatidest. Rakenduses on näha ka peatükis 4.2.2 mainitud vead. Valides jaama tunnusega "71 - Pallas", on näha hoopis 2020 aasta andmeid Ujula tänaval asunud virtuaalparkla kohta.

Vektorpaane kasutav lahendus vastab pea kõigile funktsionaalsetele nõuetele. Andmetele, mis on eeltöödeldud vektorpaanideks, ei saa rakendada erinevaid kuupäevavahemikke, kuid see-eest on jaamapõhisesee vaatesse navigeerimisel päringud minimaalse suurusega ning võtavad aega umbes 300 millisekundit. Lahendus vastab seega ka kõigile mittefunktsionaalsetele nõuetele.

4.4 Serveri kiht

SvelteKiti rakenduse alamlehed ja vaadete struktuur defineeritakse `+page.svelte` ja `+layout.svelte`[22] failides. Need failid on tavalised Svelte komponendid.

Iga alamlehe kohta saab luua ka ühe funktsiooni andmete laadimiseks. Sellised funktsioonid on defineeritud samas kataloogis asuvas `+page.ts` või `+page.server.ts` failis. `+page.ts` failis defineeritud funktsioon käivitatakse SSR ajal serveris ning CSR korral brauseris. Andmebaasist andmete pärimine või privaatsete keskkonnamuutujate nagu API võtmete kasutamine nõuab, et funktsioon käivitatakse ainult serveris. Siis tuleb kasutada `+page.server.ts` faili.

Rakenduses kasutatakse serveri funktsioonideks `+page.server.ts` ja `+layout.server.ts` faile. Kuvatõmmisel (Joonis 4) on kuvatud rakenduse kataloogipuu, kus on näha marsruudifailide kasutamist. Navigeerides otse jaama vaatesse[23] kasutab SvelteKit kõiki `+layout` faile ning jaama alamkataloogi[24] `+page` faile.

[21] https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Range_requests
[22] määrab ühise struktuuri ja väljanägemise kõikidele alamlehtedele
[23] Näiteks URL https://its.cs.ut.ee/bikeshare/67
[24] [station] kataloogis. Nurksulud määravad URL-i muutuva osa

Joonis 4. Kuvatõmmis rakenduse kataloogipuust.

Rakenduse käivitamisel luuakse ühendus andmebaasiga. Selleks kasutas autor duckdb-async[25] teeki. Arenduse lõpufaasis avalikustasid DuckDB autorid ametliku teegi Node.js keskonnas kasutamiseks[26]. Uus teek sisaldab rohkem funktsionaalsuseid ning on aktiivses täiendamises. Autor plaanib seda tulevikus kasutada.

Loodud andmebaasiühendust kasutatakse serveris väljakutsutavates funktsioonides andmete pärimiseks. Andmebaasipäringute tulemused kogutakse kokku ning tagastatakse kasutajale.

4.5 Rakenduse publitseerimine

SvelteKiti rakenduse publitseerimiseks on palju variante. SvelteKit pakub erinevaid adaptereid. Adapterid on väiksed platvormipõhised pistikmoodulid, mis kasutavad rakenduse redaktsiooni (ingl *build*), et genereerida vastavale platvormile sobivaim publikatsioon. Saadaval on mitmeid platvormipõhiseid adaptereid, näiteks adapter-vercel[27], adapter-netlify[28] ja adapter-cloudflare[29], aga ka platvormiagnostilisi adaptereid nagu adapter-node ja adapter-static.

Juhendaja soovil paigaldati rakendus its.cs.ut.ee serverisse kasutades tarkvara Docker [19]. Dockeri konteinereid kasutatakse rakenduse isoleerimiseks füüsilisest masinast, mis aitab ära hoida riistvarast või operatsioonisüsteemist tingitud erinevusi rakenduse töös. Dockeri

[25] https://www.npmjs.com/package/duckdb-async
[26] https://duckdb.org/docs/stable/clients/node_neo/overview.html
[27] https://vercel.com/
[28] https://www.netlify.com/
[29] https://pages.cloudflare.com/

konteiner ehitatakse Dockeri pildi põhjal. Dockeri pilt on standardne pakk, mis sisaldab kõiki rakenduse tööks vajaminevaid faile ja konfiguratsioone.

Dockeri pildi koostamise juhend kirjutatakse Dockerfile failis. Kõik Dockeri pildid luuakse valitud aluspildile (ingl *base image*). Valminud rakenduse ehitamiseks ja töötamiseks on vaja Node.js[30] tarkvara, seega valiti aluspildiks node:22. Pilt luuakse kahes etapis. Kõigepealt installeeritakse kõik teegid ning ehitatakse rakenduse redaktsioon. Teises etapis alustatakse uuesti baaspildist, installeeritakse ainult rakenduse tööks vajalikud teegid ning lisatakse juurde eelmises etapis ehitatud redaktsioon. Lõpuks lisatakse Dockeri pilti puhastatud andmebaasifail.

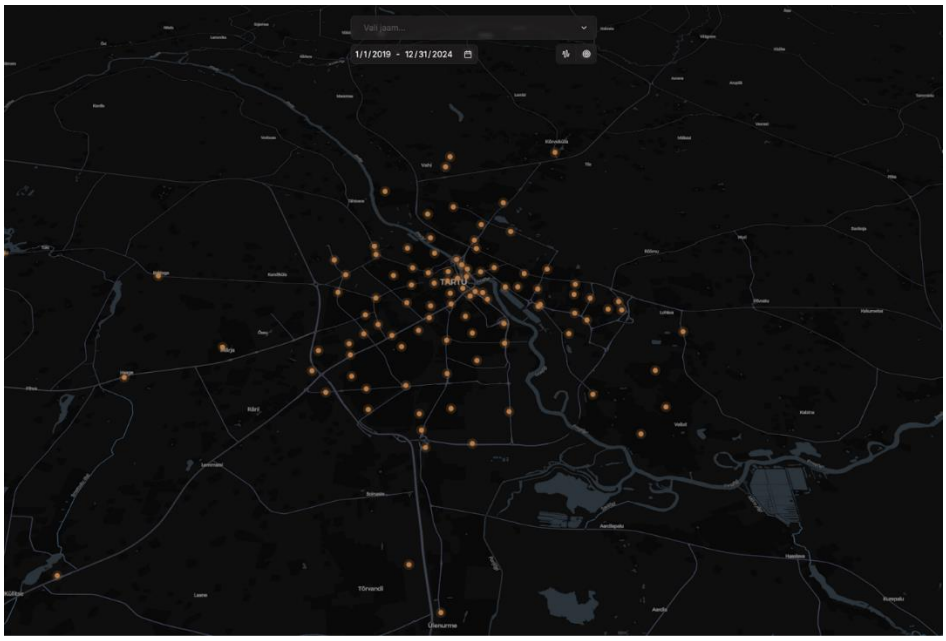
[30] https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

5. Rakenduse ülevaade ja analüüs

Selles peatükis antakse ülevaade käesolevas lõputöös loodud Tartu rattaringluse andmete visualiseerimiseks ja analüüsimiseks mõeldud koondpaneelist. Rakendus on leitav Tartu Ülikooli ITS labori internetiaadressilt https://its.cs.ut.ee/bikeshare, rakendus on kättesaadav vaid Tartu Ülikooli sisevõrgust. Rakenduse lähtekood on saadaval[31] Tartu Ülikooli Gitlabi koodihoidlas (ingl *repository*) aadressil https://gitlab.cs.ut.ee/rtalioja/smart-bike.

5.1 Rakenduse tutvustus

Rakenduse avakuval on kaart ning liitboks (Joonis 5). Filter ajavahemiku valimiseks ning nupud kaardi lisakihtide lülitamiseks on kuvatud liitboksi all. Liitboksi peale vajutades avaneb otsinguväli ning nimekiri jaamadest. Pärast jaama valimist navigeeritakse kasutaja jaamapõhisesse vaatesse. Valitud jaam muutub nimekirjas aktiivseks ning eristub teistest sinise värviga. Kuvatõmmisel (Joonis 7) on näha aktiivne otsingutermin ning selle põhjal filtreeritud jaamad, mille seas on ka valitud jaam.

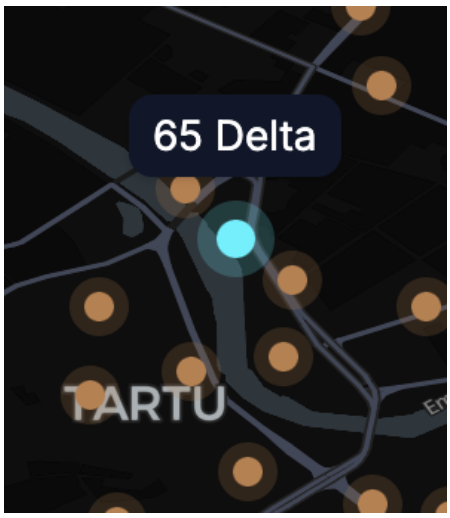


Joonis 5. Rakenduse avakuva.

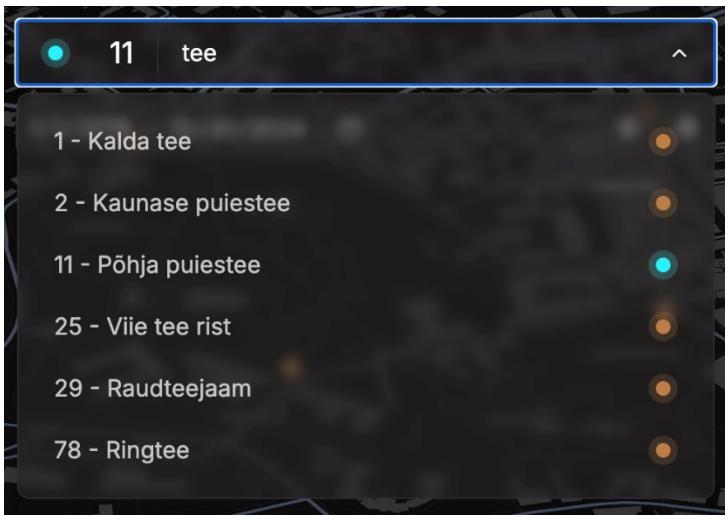
Kaardil on kuvatud asukohamärgistena kõik püsivad Tartu rattaringluse jaamad. Kursoriga märgiste peale liikudes ilmub nende kohale hüpikvihje (ingl *tooltip*) jaama nime ja tunnusnumbriga. Märgise peale vajutades navigeeritakse kasutaja jaamapõhisesse vaatesse.

[31] Koodihoidla ligipääs on piiratud. Ligipääsu saamiseks pöörduge käesoleva lõputöö autori poole.

Aktiivne märgis eristub teistest värviga. Kuvatõmmisel (Joonis 6) on näidatud olek, kui kursor hõljub valitud jaamamärgise kohal.



Joonis 6. Aktiivse jaama märgis ja hüpikvihje.



Joonis 7. Aktiivne liitboks.

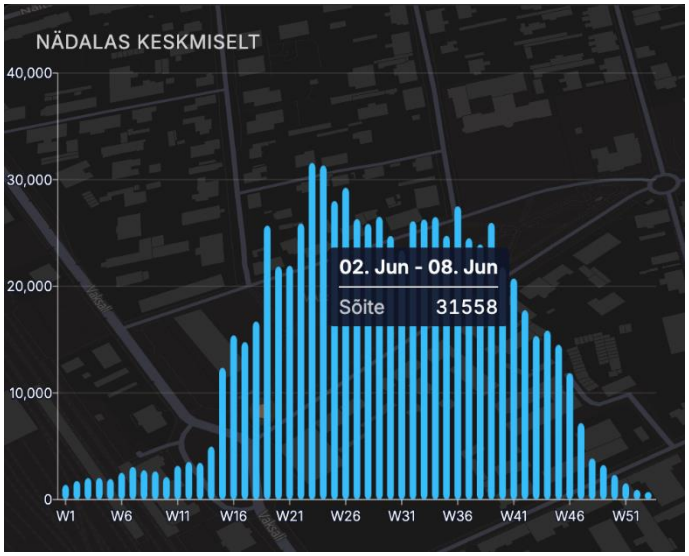
Jaamapõhises vaates (

Joonis 8) lisanduvad ekraanile külgpaneelid diagrammide ja statistiliste andmetega. Vasakpoolsel külgpaneelil on näidatud info kõikide jaamade peale kokku. Parempoolsel külgpaneelil on kasutatud valitud jaama andmeid. Kummalgi külgpaneelil on kuvatud

statistiliste väärtustena sõitude koguarv, -kestus ja -distants. Lisaks on arvutatud keskmine kestus ja distants. Jaamapõhiste andmete filtreerimiseks kasutati sõidu lähtejaama.

Joonis 8. Jaamapõhine vaade.

Kummalgi paneelil on diagrammid nädalate keskmiste sõitude arvudega ning populaarsemate jaamadega. Hõljudes kursoriga üle diagrammi kuvatakse ekraanile hüpikvihje täpsete andmetega (Joonis 9).



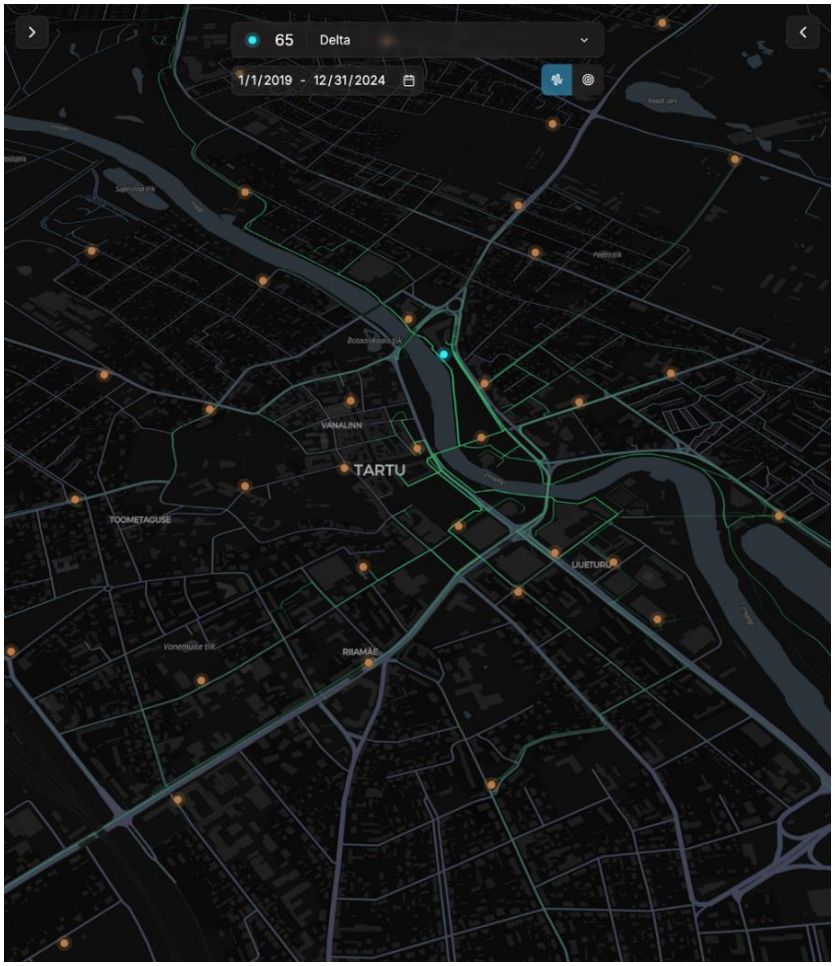
Joonis 9. Diagrammi hüpikvihje.

Liitboksi all paremal on kaks lülitit. Parempoolse lüliti aktiveerimisel kuvatakse kaardile jaamast alustatud sõitude andmete põhjal genereeritud kuumkaardi kiht (Joonis 10). Kuumkaardi värviskaala on läbipaistvast valgest kuni punaseni. Kohtades, kus alamkaart

selgelt läbi paistab puuduvad valitud jaamast alustatud sõitude kohta GPS-andmed. Asukohtades, mille peale on kuvatud läbipaistmatu tumepunane, on suur hulk GPS-andmeid. See annab ülevaate populaarsetest teelõikudest.

Joonis 10. Sisselülitatud kuumkaardi kiht.

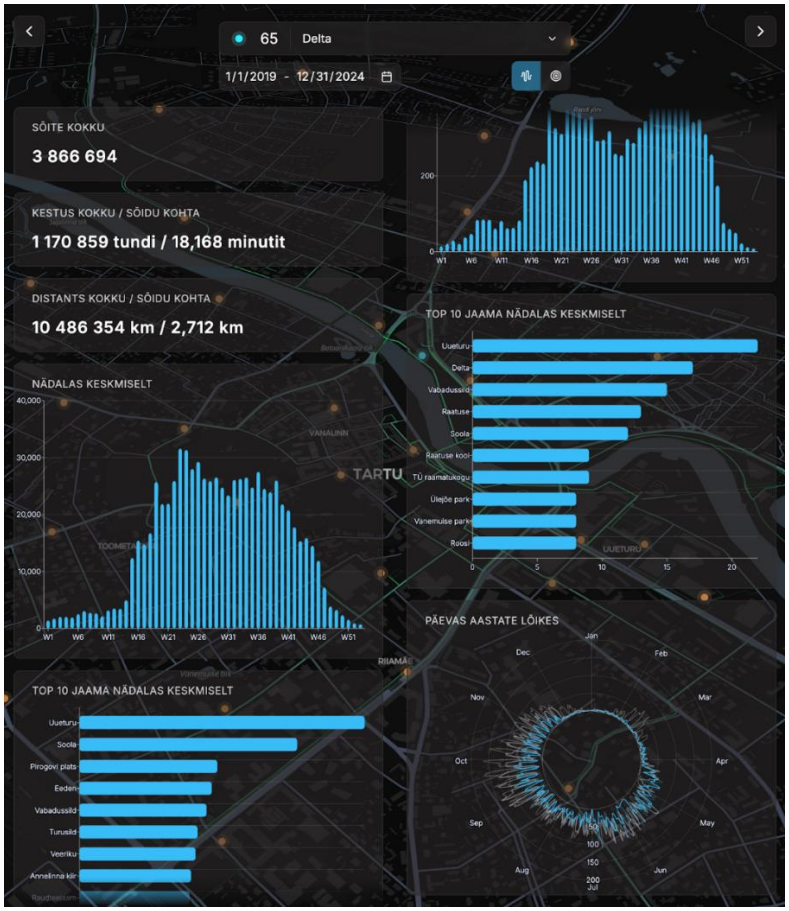
Vasakpoolse lüliti aktiivseks vajutamisel kuvatakse kaardile jaamast lähtuvate sõitude andmete põhjal genereeritud teekondade kiht (Joonis 11). Rohelise värvi tugevus näitab teekondade sagedust. Mida tumedam joon, seda sagedasem on liiklus. Teekondade kiht kasutab ainult 2020. aasta andmeid. Jaamadel, mis avati 2021. aastal või hiljem, antud teekondade kiht puudub.



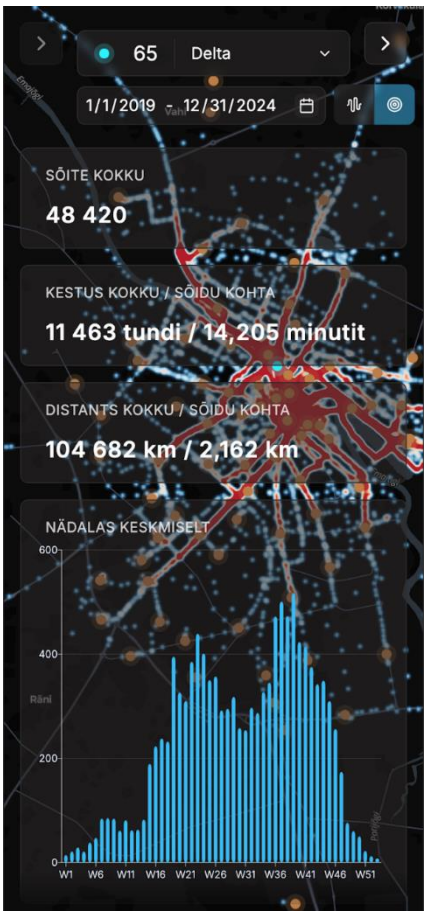
Joonis 11. Peidetud külgpaneelid ja teekondade kiht.

Rakendus on arendatud responsiivsena (ingl *responsive*). See tähendab, et koondpaneel on kasutatav erinevate suurustega ekraanidel. Kitsamal ekraanil[32] muutuvad külgpaneelid peidetavaks. Eelnev kuvatõmmis on tehtud peidetud külgpaneelidega (Joonis 11). Nurkadesse ilmuvad selleks vastavad nupud. See teeb kaardi kasutamise võimalikuks ka siis, kui külgpaneelid seda muidu kataksid (Joonis 13).

[32] näiteks pool sülearvuti ekraani või vertikaalformaadis tahvelarvuti



Joonis 13. Avatud külgpaneelid väiksel ekraanil.



Joonis 12. Avatud parempoolne külgpaneel mobiilseadmel.

Mobiilseadmetel ja teistel väga kitsastel brauseriakendel on võimalik avada korraga vaid üks külgpaneel (Joonis 12). Nupp, mis peaks avama teise külgpaneeli on sel ajal blokeeritud olekus. Selleks, et näha teist külgpaneeli tuleb esimene sulgeda.

5.2 Edasiarenduse võimalused

Kuigi kõik funktsionaalsed nõuded on täidetud, ei ole rakendus veel lõplik. Järgnevas alampeatükis toob autor välja mõned edasiarenduse võimalused.

- Filtrite lisamine, et rakendus oleks sihtgrupile kasulikum - näiteks ratta tüübi, kasutaja vanuse ning kellaaja valik.
- Diagrammide lisamine, et andmeid rohkematest vaatepunktidest uurida.
- Keele vahetamise võimalus - rakendus peaks olema kättesaadav ka eesti keelt mitte valdavatele kasutajatele.

- 2024. aasta andmestiku kasutamine, et analüüsitavad andmed oleksid võimalikud värsked.
- Kaardile sobitamise parandamine - rakendus kasutab ainult 2020. aasta andmeid ning teekonnad ei vasta reaalsusele, sest paljusid kergliikuriga läbitavaid teelõike pole sobitamise andmestikus.
- Parema kuupäeva valimise komponendi arendamine - praegune kuupäeva valimise komponent ei toeta eelvalitud kuupäevavahemikke nagu päev, nädal, kuu ja aasta, ning pika vahemiku käsitsi valimine on tülikas.
- Diagrammide kõrvale seletuste kuvamine.
- Rakenduse kasutamise õpetuse ja andmestiku kirjelduse lisamine.
- Diagrammide ajaskaala varieeruvaks tegemine, olenevalt filtritest.
- DuckDB uuele Node.js teegile üleminek.

6. Kokkuvõte

Käesoleva lõputöö eesmärk oli luua veebipõhine koondpaneel Tartu rattaringluse andmestiku visualiseerimiseks ja analüüsimiseks. Viieaastase tegevusaja jooksul on kogutud suur kogus sõidu- ja GPS-andmed, kuid seni puudus terviklik ja interaktiivne tööriist nende andmete ülevaatlikuks käsitlemiseks.

Lõputöö alguses analüüsiti olemasolevaid sarnaseid lahendusi ja püstitati loodavale rakendusele funktsionaalsed ja mittefunktsionaalsed nõuded. Peamisteks funktsionaalseteks nõueteks olid andmete filtreerimine, võtmenäitajate kuvamine ning interaktiivne kaardivaade jaamade asukohtade, sobitatud teekondade ja kuumkaartide visualiseerimiseks. Mittefunktsionaalsed nõuded puudutasid veebipõhisust, head jõudlust ja kohanduvust erinevatele ekraanisuurustele.

Rakenduse arendusel kasutati kaasaegseid veebitehnoloogiaid. Kasutajaliidese loomiseks valiti SvelteKit raamistik. Kaardipõhiseks visualiseerimiseks kasutati MapLibre GL JS teeki, Andmete haldamiseks ja päringute teostamiseks kasutati analüütilisteks töökoormusteks optimeeritud DuckDB andmebaasisüsteemi.

Arendusprotsess hõlmas rakenduse prototüübi disaini Figma tarkvaraga, andmete põhjalikku korrastamist ja puhastamist ning andmebaasi loomist. Märkimisväärne väljakutse oli suuremahuliste ruumiliste andmete (GPS-koordinaadid ja teekonnad) efektiivne visualiseerimine kaardil. Esialgne lähenemine osutus jõudluse osas ebapiisavaks. Probleem lahendati andmete eeltöötlemisega vektorpaanideks, mis tagas oluliselt kiirema kaardipõhise visualiseerimise.

Valminud rakendus vastab seatud nõuetele ning võimaldab Tartu rattaringluse andmete interaktiivset uurimist. Rakendus võimaldab analüüsida rattaringluse kasutusmustreid nii ajaliselt kui ruumiliselt, vaadata koondnäitajaid, uurida jaamapõhist statistikat ning visualiseerida sõite kaardil sobitatud teekondade ja kuumkaardina. Rakendus on publitseeritud Tartu Ülikooli sisevõrgus its.cs.ut.ee serveris Dockeri konteineri abil. Rakendus loob aluse edasisteks analüüsideks ja võimalikuks laiendamiseks tulevikus.

Viidatud kirjandus

[1] Capital Bikeshare DC, „Capital Bikeshare DC dashboard,“ [Võrgumaterjal]. Available: https://lookerstudio.google.com/s/tfiEhJwELiI. [Kasutatud 2025 mai 10].

[2] GoodGuyThor, „Bike Sharing Dashboard,“ [Võrgumaterjal]. Available: https://goodguythor-bike-sharing-dashboard.streamlit.app/. [Kasutatud 2025 mai 10].

[3] A. Cortez, J. Sánchez ja P.-P. Vázquez, „A visual tool for the analysis of usage trends of small and medium bicycle sharing systems,“ [Võrgumaterjal]. Available: https://www.sciencedirect.com/science/article/pii/S0097849322001777. [Kasutatud 2025 mai 10].

[4] R. Harris, „Svelte dokumentatsioon,“ [Võrgumaterjal]. Available: https://svelte.dev/. [Kasutatud 2025 mai 10].

[5] Stack Exchange Inc, „2024 Developer Survey - Admired and Desired,“ [Võrgumaterjal]. Available: https://survey.stackoverflow.co/2024/technology#admired-and-desired. [Kasutatud 2025 mai 10].

[6] S. Greif ja E. Burel, „State of JS 2024 - Front-end frameworks,“ Devographics, 2024 detsember 16. [Võrgumaterjal]. Available: https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/. [Kasutatud 2025 mai 10].

[7] MapLibre, „MapLibre GL JS,“ [Võrgumaterjal]. Available: https://maplibre.org/maplibre-gl-js/docs. [Kasutatud 2025 mai 10].

[8] The Khronos® Group Inc, „LOW-LEVEL 3D GRAPHICS API BASED ON OPENGL ES,“ [Võrgumaterjal]. Available: https://www.khronos.org/webgl/. [Kasutatud 2025 mai 10].

[9] D. Imfeld, „Svelte MapLibre,“ [Võrgumaterjal]. Available: https://svelte-maplibre.imfeld.dev/. [Kasutatud 2025 mai 10].

[10] Mierune Inc., „Svelte Maplibre GL,“ [Võrgumaterjal]. Available: https://github.com/MIERUNE/svelte-maplibre-gl?tab=readme-ov-file. [Kasutatud 2025 mai 10].

[11] M. Raasveldt ja H. Mühleisen, „DuckDB Documentation,“ [Võrgumaterjal]. Available: https://duckdb.org/docs/stable/index. [Kasutatud 2025 mai 10].

[12] Interaction Design Foundation, „IxDF. "What is Glassmorphism?",“ [Võrgumaterjal]. Available: https://www.interaction-design.org/literature/topics/glassmorphism. [Kasutatud 2025 mai 10].

[13] T. G. Lopes, „Melt UI,“ [Võrgumaterjal]. Available: https://www.melt-ui.com/docs/introduction. [Kasutatud 2025 mai 10].

[14] S. Lynch, „LayerChart,“ [Võrgumaterjal]. Available: https://www.layerchart.com/. [Kasutatud 2025 mai 10].

[15] M. Keller, „Layer Cake,“ [Võrgumaterjal]. Available: https://layercake.graphics/. [Kasutatud 2025 mai 10].

[16] Cloud-Native Geospatial Foundation, „Cloud+Optimized Geospatial Formats Guide + PMTiles,“ 2023. [Võrgumaterjal]. Available: https://guide.cloudnativegeo.org/pmtiles/intro.html. [Kasutatud 2025 mai 10].

[17] E. FIscher, „Tippecanoe,“ [Võrgumaterjal]. Available: https://github.com/felt/tippecanoe. [Kasutatud 2025 mai 10].

[18] Protomaps, „PMTiles Concepts,“ [Võrgumaterjal]. Available: https://docs.protomaps.com/pmtiles/. [Kasutatud 2025 mai 10].

[19] Docker Inc, „Dockerdocs - Get started,“ [Võrgumaterjal]. Available: https://docs.docker.com/get-started/. [Kasutatud 2025 mai 10].

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Rasmus Talioja ,

(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Veebipõhine koondpaneel Tartu rattaringluse andmete visualiseerimiseks ja analüüsimiseks ,

   (lõputöö pealkiri)

mille juhendaja(d) on Helen Tera ,

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;
3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;
4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Rasmus Talioja

15.05.2025