Projekt rt21

David Slatinek **Marcel Iskrač** Marko Hiršel

Junij 2021

Povzetek

Ta dokument opisuje infrastrukturo projekta rt21. Najprej opišemo vzpostavitev strežnika, za katero platformo smo se odločili in alternative. Nato opišemo izdelavo podatkovne baze, izbrano tehnologijo in alternative. Zatem predstavimo izdelavo API-ja in Android aplikacije, uporabljene tehnologije in druge možnosti. Na koncu predstavimo izdelavo spletne strani, uporabljeno tehnologijo in druge pristope.

1 Vzpostavitev strežnika

Strežnik smo vzpostavili na platformi Heroku. Na njem tečeta API[1], katerega storitve uporabljata Android aplikacija in spletna stran, ki prav tako gostuje na tej platformi. API in spletna stran tečeta znotraj Docker zabojnika [1].
Razlogi za platformo Heroku:

- Zastonj gostovanje.
- Enostavna konfiguracija.
- Enostavna uporaba.
- Podpora https protokola.

Druge možnosti:

- Lasten strežnik.
- AWS[2].
- Linode.

Pri vzpostavitvi lastnega strežnika bi pridobili največ svobode in ne bi imeli kakšnih omejitev (npr. pri Heroku je delovanja strežnika omejeno na 550 ur na mesec), bi pa vzpostavitev takšnega strežnika trajala najdlje časa, saj bi morali poskrbeli za vse vidike, ki so navedeni v tabeli 1.

Druga možnost bi bila gostovanje na platformi AWS. Vzpostavitev strežnika na tej platformi bi prav tako bila izjemno hitra in enostavna, slabost tega je pa, da je platforma zelo omejena glede zastonj možnosti, saj se večinoma osredotočajo na velika podjetja. V primeru izbire te opcije bi prav tako morali pri registraciji navesti številko kreditno kartico.

Zadnja možnost je platforma Linode, ki je cenejša alternativa AWS, a kljub temu so zelo omejeni glede zastonj gostovanj.

Zadolžitev: **David Slatinek.**

[1] Application Programming Interface.
[2] Amazon Web Service.

Konfiguracija statičnega IP-naslova
Namestitev Dockerja
Namestitev strežnika, npr. Apache, Nginx
Namestitev sistema za podatkovno bazo
Konfiguracija odpiranja vrat
Konfiguracija https protokola
Konfiguracija DNS strežnika
Namestitev in konfiguracija ipban-a
Namestitev in konfiguracija orodja za spremljanje sistema

Tabela 1: Vzpostavitev lastnega strežnika.

2 Izdelava podatkovne baze

Načrtovanja podatkovne baze se je treba lotiti z veliko skrbjo, saj v nasprotnem primeru lahko pride do veliko težav [2], na primer:

- Podatkovna baza vsebuje premalo oziroma preveč podatkov.
- Težavna razširitev.
- Težavno vzdrževanje.

Za podatkovno bazo smo izbrali NoSQL tip baze, natančneje MongoDB. Za gostovanje podatkovne baze smo uporabili MongoDB Atlas, ki ima različne nivoje gostovanj, izbrali smo zastonj verzijo. V bazi shranjujemo podatke o uporabniku, njegovih vožnjah, lokacijah teh voženj in podatke o prometnih znakih.
Razlogi za izbor MongoDB [3, 4]:

- Popularnost - veliko dokumentacije, primerov.
- Hiter razvoj.
- Enostavna razširitev.
- Brez kompleksnih združevanj.
- Enostavna struktura - JSON[3].

Druge možnosti:

- MySql.
- T-SQL.
- Cassandra.

MySQL in T-SQL sta relacijski bazi, Cassandra spada pod stolpčne NoSQL baze. Največ izkušenj imamo z MySQL in T-SQL, a kljub vsemu smo se odločili za NoSQL tip baze, razlogi so navedeni zgoraj.

Z uporabo klasičnih podatkovnih baz bi s pravilnim načrtovanjem onemogočili oziroma omejili redundanco podatkov, vendar bi nadaljnji razvoj bil težavnejši kot pri NoSQL bazah. Prav tako so NoSQL baze namenjene velikim količinam podatkov, ki bi nastali pri velikem številu uporabnikov naše aplikacije.

Zadolžitev: **David Slatinek.**

[3] JavaScript Object Notation.

3 Izdelava API-ja

API je narejen s programskim jezikom python oziroma z njegovim ogrodjem, flask in temelji na arhitekturi REST[4], ki vsebuje HTTP [5] metode. API služi kot vmesni člen med odjemalci in podatkovno bazo. Z njegovo uporabo omejimo nedovoljen dostop do baze, prav tako je razvoj čelnih aplikacij enostavnejši, saj se razvijalci, ki delajo na tem, ne ukvarjajo s pridobivanjem podatkov iz baze, ampak jih pridobijo v določeni obliki, nato pa jih uporabijo v nadaljnjem razvoju.

Razlogi za izbor flask-a:

- Enostaven razvoj.
- Preprosto vzdrževanje.
- Izkušnje s tem ogrodjem.

Podatki se vračajo v obliki JSON, druga možnost bi bila XML. Podpiramo vse operacije CRUD:

- C - Create - ustvarjanje novih zapisov.
- R - Read - pridobivanje podatkov.
- U - Update - posodabljanje podatkov.
- D - Delete - brisanje podatkov.

Druge možnosti:

- Ogrodje django - python.
- Node.js.
- Ruby on Rails.

Za izdelavo API-ja lahko uporabimo bolj ali manj večino modernejših programskih jezikov oziroma ogrodij, ki temeljijo na teh jezikih, so pa zato nekateri primernejši za to nalogo. Med te možnosti spadajo django, Node.js in Ruby on Rails.

Django je zelo popularno python ogrodje, ki se uporablja za izdelavo spletnih storitev. Njegova glavna prednost je množica razširitev, ki naredijo ogrodje zelo močno, a tudi kar kompleksno. Druga možnost bi bila uporaba tehnologije Node.js. Zadnja možnost bi bila uporaba Ruby on Rails.

Vmesnik je uporaben, namenjen svojemu namenu, zanesljiv, razširljiv in varen. Kar se dotika vidika varnosti, sistem vsebuje naslednje varnostne mehanizme [1, 5]:

- Identifikacija - API ključ.
- Šifriranje - TLS[6].

Varnostni vidik pri uporabi API-ja je zelo pomemben. Uporabo API-ja želimo omejiti na samo določene uporabnike, npr. samo tiste, ki imajo določen ključ. Da preprečimo škodljivo uporabo API-ja, je njegova uporaba mogoča samo z uporabo ustreznega ključa. Prav tako platforma Heroku (gostovanje API-ja) poskrbi za šifriranje podatkov - protokol https[7].

Zadolžitev: **David Slatinek.**

[4] REpresentational State Transfer.
[5] HyperText Transfer Protocol.
[6] Transport Layer Security.
[7] Hypertext Transfer Protocol Secure.

4 Izdelava Android aplikacije

Android aplikacijo smo razvili v razvojnem okolju Android Studio ter s programskem jeziku java. To možnost smo izbrali predvsem zaradi predhodnega poznavanja okolja in programskega jezika ter zaradi preproste uporabe komponent naprave. Težava tega je v prenosljivosti aplikacije na iOS platformo.
Aplikacija omogoča prijavo in spremembo gesla uporabniškega računa. Glavna funkcionalnost aplikacije je zajem slik in podatkov iz senzorjev ter pošiljanje teh na strežnik.

S tem, da smo za razvoj mobilne aplikacije izbrali nativni način, smo pridobili naslednje [6]:

- Boljše grafične zmožnosti.
- Dostop aplikacije preko storitve Google Play.
- Podpora za uporabo komponent naprave.
- Hitrost.

Kljub vsem prednostim, smo na podlagi te odločitve pridobili nekaj pomanjkljivosti [6]:

- Slaba prenosljivost - SDK[8], Android OS[9].
- Daljši čas razvoja.
- Nestabilnost platforme.
- Večji strošek vzdrževanja.
- Omejen nadzor.

Alternativne možnosti:

- React Native.
- Ionic.
- Fluter.

Alternativne možnosti so primernejše za razvoj prenosljivih aplikacij in pa nasploh za razvoj lepših in modernejših uporabniških vmesnikov.

V primeru izbire modernejših načinov izdelave mobilne aplikacije bi to imelo naslednje prednosti [6]:

- Neodvisnost od platforme.
- Enostavno vzdrževanje.
- Hiter razvoj.

Bi pa ta odločitev imela določene slabosti, predvsem slabo podporo za uporabo komponent naprave in omejeno grafiko ter manjšo hitrost v primerjavi z nativno aplikacijo [6].

Zadolžitev: **Marcel Iskrač, Marko Hiršel**.

[8]Software Development Kit.
[9]Operacijski sistem.

5 Izdelava spletne strani

Spletna stran smo ustvarili s pomočjo JavaScript knjižnice React, HTML in CSS ter Boostrap knjižnice za enostavno oblikovanje elementov. React smo uporabili za postavitev in klic komponent aplikacije in za komunikacijo med API-jem in spletno stranjo. React smo izbrali zaradi preproste izdelave uporabniškega vmesnika in zaradi predhodnega poznavanja.

Prednosti React-a [7, 8]:

- Dinamičen prenos podatkov brez potrebe po ponovnem nalaganju strani.
- Gradnja aplikacijskega dela na odjemalčevi strani.
- Zmanjšana obremenjenost zalednega strežnika (izvajanje logike iz zaledne strani na odjemalčevo stran).
- Odzivnost aplikacije je neodvisna od obremenjenosti zalednega dela.
- Ustvarimo hitro, odzivno aplikacijo SPA[10] na strani odjemalca.
- Enostavno vzdrževanje - uporaba komponent.

Slabosti [8]:

- Daljši prvi zagon spletne aplikacije.

Alternativne možnosti:

- Node.js.
- Angular.
- Vue.js.

Alternativne možnosti, enako kot React, temeljijo na programskem jeziku JavaScript, kar pomeni, da so to le različna ogrodja, ki olajšajo razvoj spletnih aplikacij.
Node.js je ogrodje, ki izvaja JavaScript kodo na strani spletnega strežnika. Uporablja neblokiran vhodno/izhodni model, ki temelji na dogodkih, kar pomeni, da je lahek in učinkovit [9].
Angular je ogrodje, ki se osredotoča na podporo za izdelavo velikih aplikacij in SPA [10] in temelji na jeziku TypeScript.
Vue.js je JavaScript ogrodje, ki se uporablja za gradnjo spletnih vmesnikov in enostranskih aplikacij. Vue.js se ne uporablja le za spletne vmesnike, temveč tudi za razvoj namiznih in mobilnih aplikacij z ogrodjem Electron [11].

Zadolžitev: **Marcel Iskrač.**

6 Varnost in prenosljivost

Varnost podatkov je pri vsakem projektu izjemnega pomena. Za ta namen pri vseh delih projekta uporabljamo protokol https, uporabniška gesla pa so v podatkovni bazi shranjena kot hash-i. Z uporabo Dockerja dosežemo prenosljivost.
Tabela 2 prikazuje varnost projekta in prenosljivost določenega dela.

[10] Single Page Application.

Del projekta	Varnost	Prenosljivost
Strežnik	Ponudnik, https	Da
Baza	Ponudnik, računi, https	Da
API	Ponudnik, API ključ, https	Da
Android aplikacija	Https	Android platforma
Spletna stran	Ponudnik, https	Da

Tabela 2: Varnost in prenosljivost projekta.

Literatura

[1] David Slatinek, Marcel Iskrač in Marko Hiršel. "Pregled rezultatov sprinta 1". 15. maj 2021. (Pridobljeno 21. 5. 2021).

[2] *Database Planning Guide*. Feb. 2001. URL: `https://healtorture.org/sites/healtorture.org/files/Database%20Planning%20Workbook.pdf` (pridobljeno 16. 11. 2020).

[3] *MongoDB - Advantages*. URL: `https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm` (pridobljeno 16. 11. 2020).

[4] David Slatinek. "Načrt razvoja programske opreme za projekt Inteligentni urnik - opredelitev, časovni načrt in razdelitev projekta". 28. okt. 2020. (Pridobljeno 15. 3. 2020).

[5] Maksym Churylov. *How to build an API (without spending a fortune and going crazy)*. URL: `https://www.mindk.com/blog/how-to-build-an-api/` (pridobljeno 16. 11. 2020).

[6] doc. dr. Tomaž Kosar. *Razvoj aplikacij za internet 2021/2021: Hibridne mobilne aplikacije*. URL: `https://estudij.um.si/course/view.php?id=11324` (pridobljeno 4. 6. 2021).

[7] doc. dr. Tomaž Kosar. *Razvoj aplikacij za internet 2021/2021: React*. URL: `https://estudij.um.si/course/view.php?id=11324` (pridobljeno 5. 6. 2021).

[8] doc. dr. Tomaž Kosar. *Razvoj aplikacij za internet 2021/2021: SPA, Deep linking*. URL: `https://estudij.um.si/course/view.php?id=11324` (pridobljeno 5. 6. 2021).

[9] doc. dr. Tomaž Kosar. *Razvoj aplikacij za internet 2021/2021: Node.js*. URL: `https://estudij.um.si/course/view.php?id=11324` (pridobljeno 5. 6. 2021).

[10] doc. dr. Tomaž Kosar. *Razvoj aplikacij za internet 2021/2021: Angular*. URL: `https://estudij.um.si/course/view.php?id=11324` (pridobljeno 5. 6. 2021).

[11] *The Good and the Bad of Vue.js Framework Programming*. 11. sep. 2019. URL: `https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/` (pridobljeno 5. 6. 2021).

A Terminski načrt

A.1 Vzpostavitev strežnika

Predmet: Sistemska administracija.

- ☑ Gostovanje na platformi herokuapp
  - ☑ Registracija na platformi herokuapp
- ☒ Lasten strežnik
  - ☒ Konfiguracija statičnega IP naslova
  - ☒ Namestitev dockerja
  - ☒ Namestitev strežnika, npr. Apache, Nginx
  - ☒ Namestitev sistema za podatkovno bazo

- ☒ Konfiguracija port forwarda
- ☒ Konfiguracija https protokola
- ☒ Konfiguracija DNS strežnika
- ☒ Namestitev in konfiguracija ipban-a
- ☒ Namestitev in konfiguracija orodja za spremljanje sistema

☑ Izgradnja docker slike

☑ Namestitev flask ogrodja

☑ Ustvarjanje novega projekta za API gostovanje

☑ Namestitev projekta

A.2 Izdelava podatkovne baze

Predmet: Razvoj aplikacij za internet.

☑ Registracija na platformi MongoDB Atlas

☑ Načrtovanje NoSQL baze

☑ Seznanitev s pridobivanjem podatkov

☑ Definiranje shem

☑ Definiranje imen in podatkovnih tipov

☑ Ustvarjanje zbirk

A.3 Izdelava API-ja

Predmet: Razvoj aplikacij za internet, sistemska administracija.

☑ Namestitev programa znotraj dockerja

☑ Podpora https protokola

☑ Podpora za API ključe

☑ Različne metode

- ☑ Pridobivanje podatkov
- ☑ Vstavljanje podatkov
- ☑ Posodabljanje podatkov
- ☑ Brisanje podatkov

A.4 Izdelava Android aplikacije

Predmet: Uvod v platformno odvisen razvoj aplikacij, osnove računalniškega vida.

- ☑ Prijava in registracija
- ☑ Zajemanje slike iz kamere
- ☑ Zaznavanje prometnih znakov
- ☒ Spremljanje hitrosti
- ☑ Uporaba GPS za spremljanje lokacije
- ☒ Spremljanje smeri vožnje - vožnja med črtami
- ☒ Zaznavanje tresljajev za določanje kvalitete vožnje
- ☑ Prikaz osnovnih podatkov
- ☑ Zbiranje statističnih podatkov

A.5 Izdelava spletne strani

Predmet: Razvoj aplikacij za internet, sistemska administracija.

- ☑ Izgradnja docker slike
- ☑ Namestitev na heroku
- ☑ Podpora https protokola
- ☑ Glavna stran
- ☒ Info stran
- ☑ Prijava
- ☑ Registracija
- ☑ Vizualizacija podatkov