Poročilo - Projekt Nepremičnine

RSO 2024/2025

Krištof Zupan – Skupina: Samostojno 15

GitHub Repozitorij: https://github.com/FRI-RSO-2024-Skupina-15/Projekt_nepremicnine

Dostop do aplikacije:

- http://72.146.60.142/

Dostop do OpenApi dokumentacije:

- http://72.146.60.142/api/properties/docs/
- http://72.146.60.142/api/images/docs/

Kratek opis:

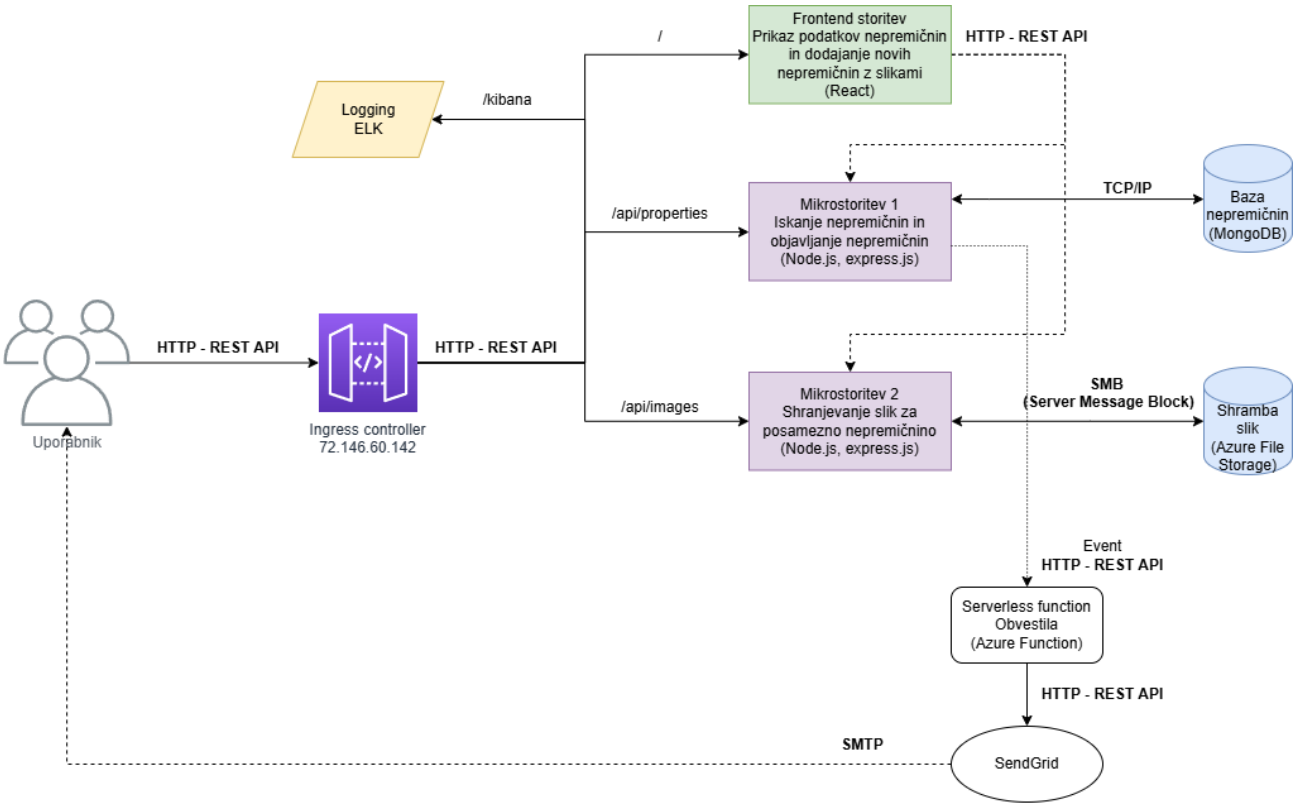
Projekt Nepremičnine se je osredotočil na vzpostavitev mikrostoritev za aplikacijo za iskanje nepremičnin. Glavni cilji aplikacije so bili razumeti delovanje in razvoj za storitve v oblakih. Za to je bila ideja, da se razvije dve mikrostoritvi. Prva je osredotočena na shranjevanje in pridobivanje nepremičnin iz podatkovne baze. Druga mikrostoritev je bila osredotočena na shranjevanje slik za vsako nepremičnino. Dodatno je bila implementirana še »serverless« funkcija za obveščanje uporabnika, ko je dodana nova nepremičnina. Tukaj je bil glavni cilj pokazati koncepte, ki so pomembni pri razvoju storitev v oblaku. Za shranjevanje nepremičnin se uporablja MongoDB baza, za shranjevanje slik se pa uporablja »Azure File Storage«.

Ogrodje in razvojno okolje

Za Projekt Nepremičnine so uporabljene slednja okolja in ogrodja. Za oblačne storitve je bil uporabljen Microsoft Azure. Ustvarjeno je bilo Azure Kubernetes Service (AKS), kjer je aplikacija nameščena. Za razvoj posameznih mikrostoritev je bil uporabljen Node.js/Express.js. Za shranjevanje podatkov v podatkovni bazi je uporabljen MongoDB in za shranjevanje slik nepremičnin je uporabljen Azure File Storage. Aplikacija je shranjena na GitHubu, kjer se tudi izvaja CI/CD za avtomatsko postavljanje aplikacije z uporabo GitHub Actions. Vsaka storitev je tudi shranjena kot slika z uporabo Docker-ja. Za pošiljanje elektronskih sporočil ko je dodana nova nepremičnina se uporablja zunanji API SendGrid. Za proženje te storitve se uporablja Azure functions. Za prikaz podatkov se uporablja spletni vmesnik narejen z ogrodjem React.js.

Dodatne informacije so v tehnični dokumentaciji.

Shema arhitekture



Seznam funkcionalnosti mikrostoritev

1. Mikrostoritev nepremičnine:

- Imamo 3 CRUD operacije za dodajanje nepremičnine, iskanje med nepremičninami in brisanje nepremičnin.
    - GET /api/properties
      Omogoča iskanje med nepremičninami z uporabo različnih filtrov.
    - POST /api/properties
      Omogoča dodajanje nove nepremičnine v MongoDB bazo
    - DELETE /api/properties/:id
      Omogoča brisanje nepremičnine iz baze, ki ima določen id.

  Ko se sproži dodajanje nove nepremičnine se tudi sproži klic na »serverless« funkcijo, ki sproži pošiljanje elektronske pošte kot obvestilo o dodani novi nepremičnini.

  Za vse filtre, ki jih omogoča GET, poglejte dokumentacijo ali Swagger na naslovu /api/properties/docs.

2. Mikrostoritev slike:

- Imamo tudi 3 CRUD operacije za dodajanje, iskanje in brisanje slik za posamezno nepremičnino.

- GET /api/images/property/:id
  Omogoča pridobitev podatkov o sliki in url, kjer se jo pridobi za izris na spletnem vmesniku.
- POST /api/images/property/:id
  Omogoča dodajanje slik za nepremičnino. Slike je lahko največ 10 in morajo biti manjše od 5 MB.
- DELETE /api/images/:id
  Omogoča brisanje slike z določenim id-jem.

Iz pridobljenih podatkov dobimo url naslove za slike, ki so od neke nepremičnine. Tako lahko na koncu uporabimo ta url za izris na spletni strani z dostopom preko te povezave: /api/images/uploads/<image>.

Spet za bolj podrobne informacije o strukturi poglejte tehnično dokumentacijo ali Swagger na naslovu /api/images/docs.

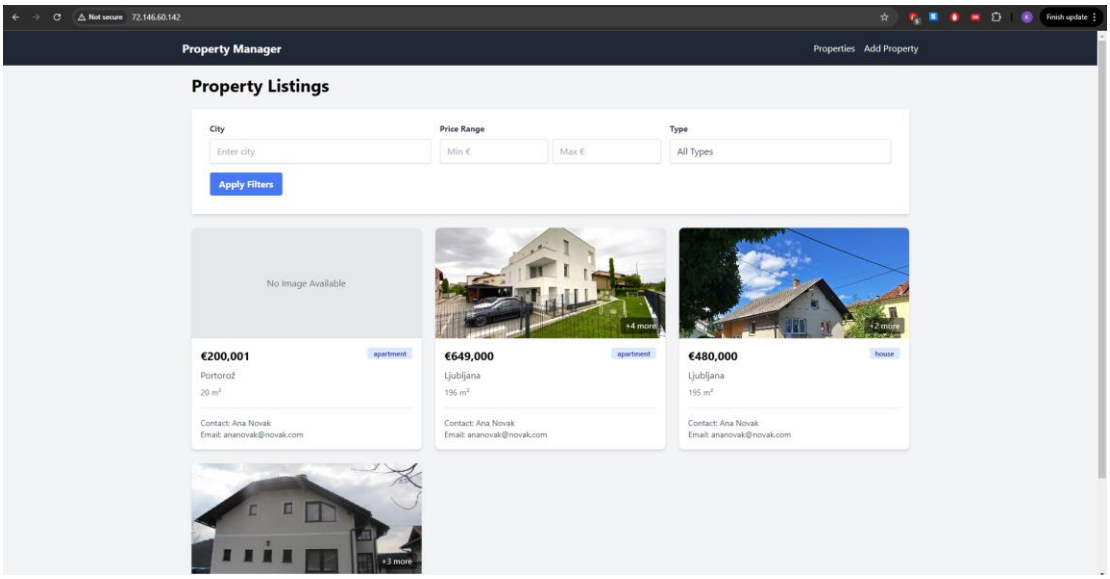
3. Storitev za uporabniški vmesnik

   Ta storitev se uporablja za prikaz nepremičnin, kjer je tudi možno filtrirati podatke, glede na podatke o nepremičnini. Ta stran se nahaja na direktno na IP naslovu od »ingress« krmilnika. Druga podstran, ki je namenjena za dodajanje nepremičnin se nahaja na /add. Tam se shranijo podatki o nepremičnini in se dodajo tudi slike nepremičnine.
   Storitev je narejena v React ogrodju za spletne vmesnike.

Primeri uporabe

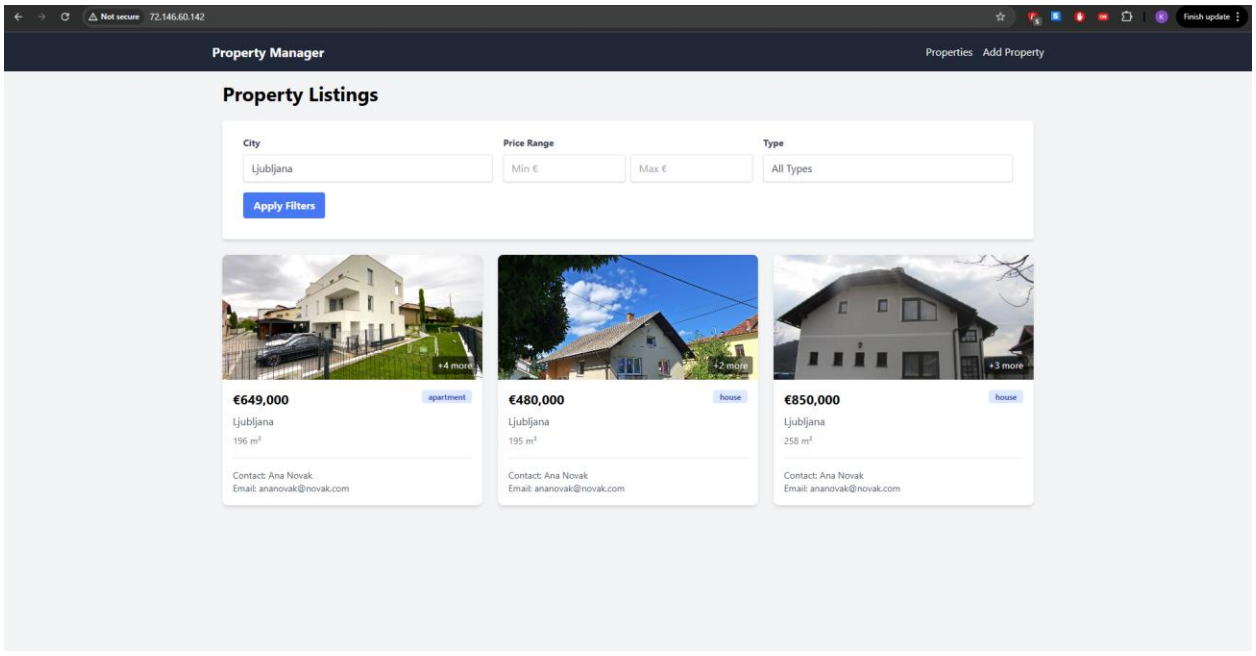
1. <u>Odprtje spletnega vmesnika</u>

   Ko se odpre spletni vmesnik se prikažejo vse nepremičnine, ki so shranjene v bazi. Najprej se pokliče klic za nepremičnine po tem pa se za vsako nepremičnino še izvede klic za pridobivanje slik. Nato se uporabi url za posamezno nepremičnino, ki je pridobljen iz klica za slike.



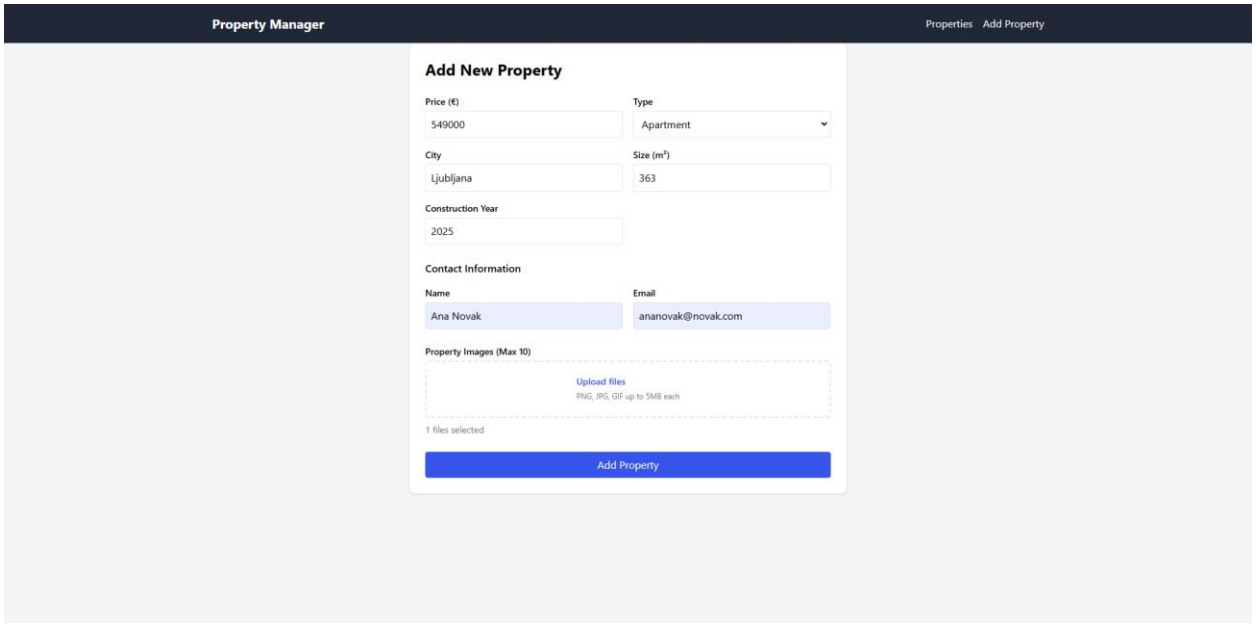
2. Filtriranje nepremičnin

V spodnjem primeru smo filtrirali po imenu mesta in dobili rezultate za to. Lahko bi mešali filtre in bi dobili rezultate, kot bi bilo predvideno. Tukaj se uporabi enak sistem kot pri prejšnjem primeru.

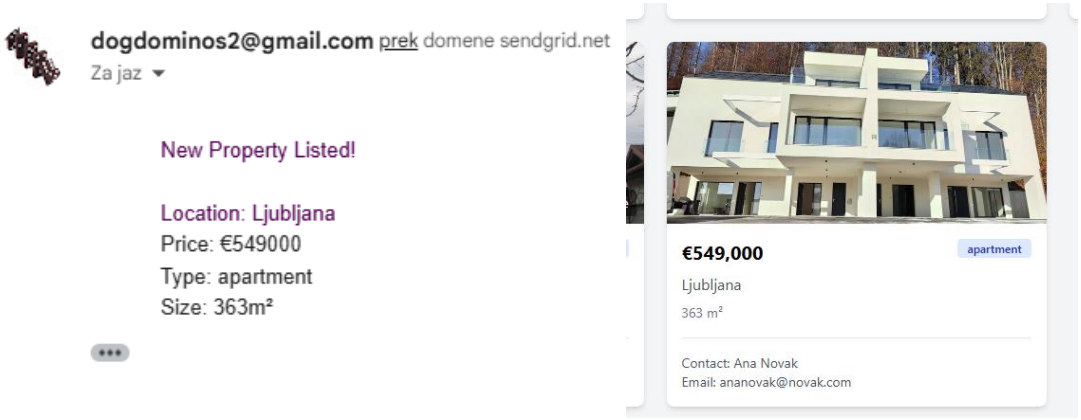


3. Dodajanje nepremičnine

V tem primeru se izvede več stvari. Najprej se izvede klic na mikrostoritev za dodajanje nepremičnin. Med tem se sproži klic za oddajo obvestila o novi nepremičnini, ki jo izvede Azure funkcija. Na koncu se pa še izvede klic na drugo mikrostoritev za dodajanje slik od nepremičnine.



Na koncu dobimo na email obvestilo in vidimo na prikazu nepremičnin, da je bila nova nepremičnina dodana.

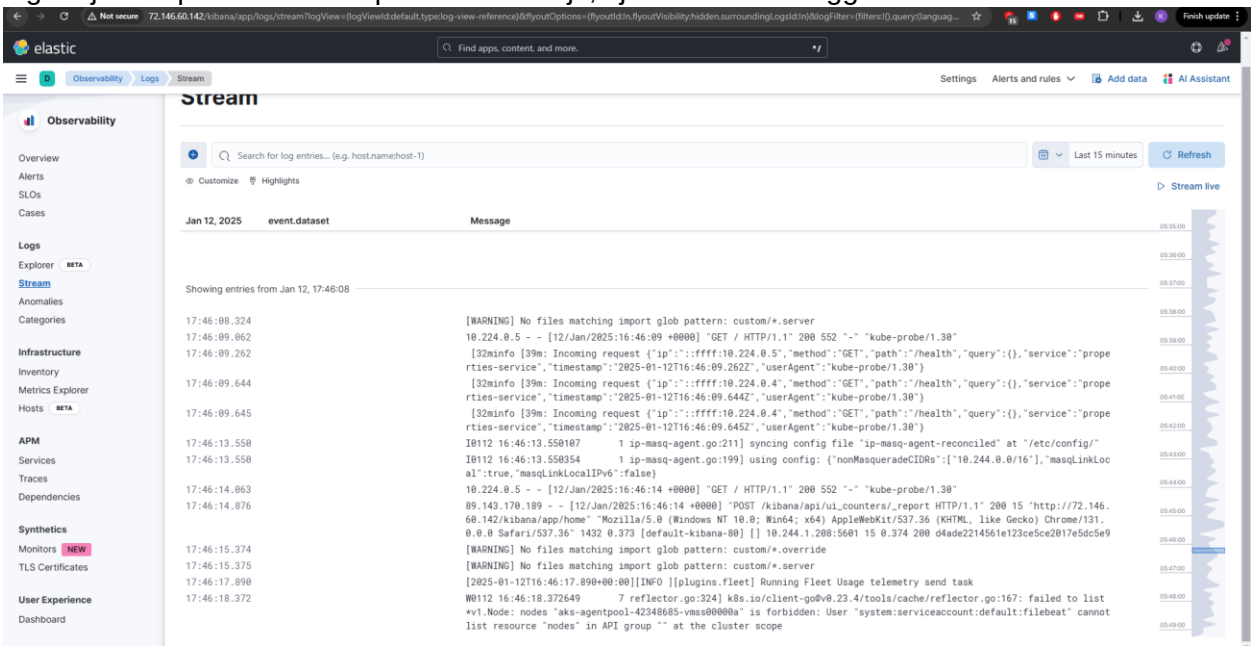


Seznam opravljenih/vključenih osnovnih in dodatnih projektnih zahtev

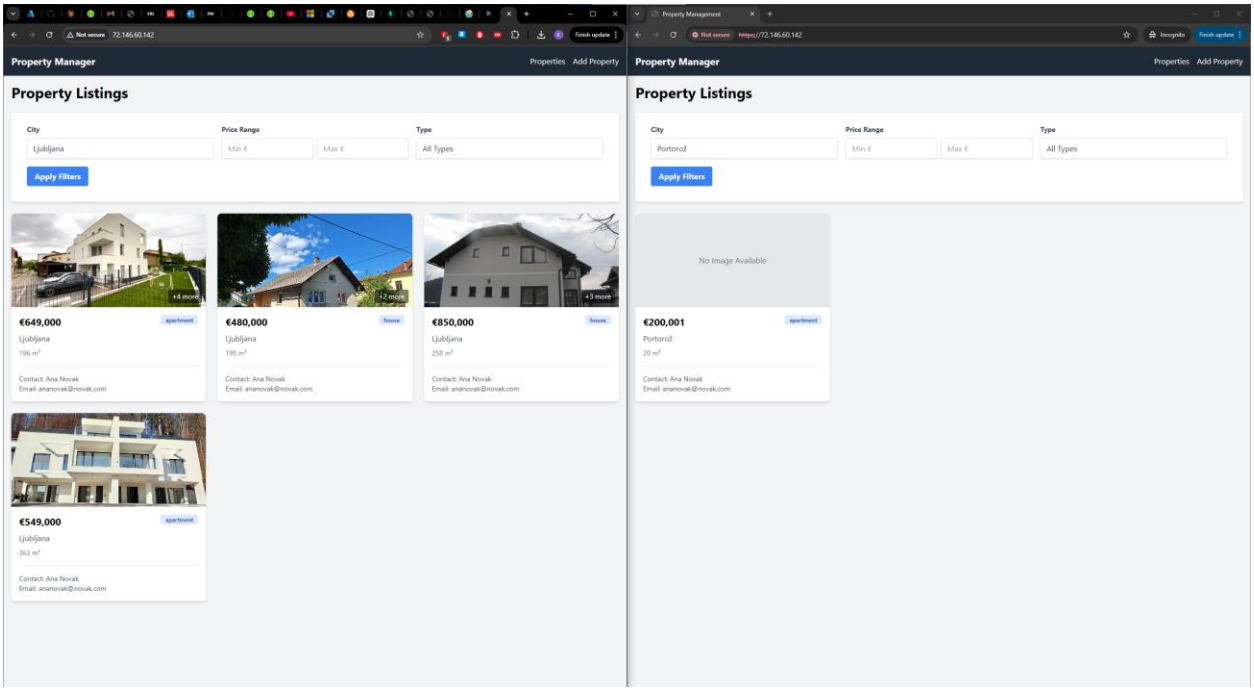
1. Repozitorij
   Na github je bil dodan repozitorij (url je zgoraj). Dodani sta dve .md datoteki za osnovno razlago je README in ena za tehnično dokumentacijo. Narejena sta dve razvejitvi za razvoj »main« in »dev«.
2. Mikrostoritve
   Opisane so že zgoraj v poročilu. Uporabljal se je VSC. Vsaka mikrostoritev se je naredilo v Docker »image« in potem namestilo z uporabo svoje Kubernetes manifest YAML datoteke. Seveda mikrostoritve tudi komunicirajo z bazo podatkov. Narejene so z uporabo Node.js in Express okolja za razvoj.
3. Dokumentacija
   Tehnična dokumentacija je svoja .md datoteka, kjer so notri navedene glavne lastnosti celotne aplikacije.
4. Dokumentacija API
   Za dokumentacijo api-ja se je dodal tudi swagger. Na voljo je za vsako mikrostoritev posebej na url-ju /docs. Notri so zapisani podatki za vsak možen klic. Na voljo je opis podatkov in struktura podatkov. Možno je tudi testirati delovanje preko swagger vmesnika.
5. Cevovod CI/CD
   Za to se je uporabljal GitHub Actions. Ko se koda doda v vejo »main« se avtomatično sproži postopek, kjer se najprej za vsako mikrostoritev naredi Docker »build« in nato se avtomatično posodobi v kubernetes gruči.
6. Namestitev v oblak
   Celotna aplikacija je nameščena v Microstoft Azure oblak. Uporablja se več storitev na tem portalu. Glavna za naš originalen projekt je AKS. Tam je aplikacija javno dostopna na zgoraj omenjenem IP naslovu. Uporabljen je bil Azure CLI in kubectl za upravljanje AKS.
7. »Serverless« funkcija
   Za to se je uporabila Azure Functions storitev, kamor se je naložila funkcija, ki se sproži ko je dodana nova nepremična v bazo podatkov. Funkcija sproži na zunanjem API-ju

(SendGrid), da se sproži pošiljanje elektronske pošte. To je tudi opisano v zadnjem primeru v prejšnjem poglavju.

8. Preverjnaje zdravja
   Za vsako mikrostoritev je bila dodana »/health« pot, ki prikaže delovanje mikrostoritve. Ko se sproži klic se preveri ali ima mikrostoritev povezavo do baze podatkov. Kot drugi del pa je bilo dodano tudi v kubernetes manifest livenessProbe in readinessProbe, ki preverjata ali je storitev v redu. V primeru, da nekaj ni v redu jo na novo zaženeta.
9. Zunanji API
   Kot je bilo že prej omenjeno se uporablja zunanji API za pošiljanje elektronskih sporočil. Uporablja se storitev SendGrid, ki ob dodani novi nepremičnini pošlje elektronsko sporočilo o podatkih o novi nepremičnini.
10. Grafični vmesnik
    Kot je že vidno v zgornjih primerih sta narejeni dve strani za ogled nepremičnin in za dodajanje nepremičnin. Uporabljalo se je ogrodje React za izdelavo spletnega vmesnika.
11. Centralizirano zbiranje dnevnikov
    Dodan je bil ELK stack, ki se ga da dostopati preko ingress poti /kibana. Implementacija je bila narejena preko manifestov .YAML, kjer se je definiralo različne datoteke za vzpostavitev Stack-a. Imamo YAML datoteke za elasticsearch, filebeat in kibana. Za logiranje se uporabi winston paket za node.js, kjer se definira logger.



12. Ingress Controller
    Narejen je bil Ingress controller, kjer so na voljo vse mikrostoritve in tudi spletni vmesnik. Narejen je bil z definiranjem manifesta ingress.yaml. Kjer so definirane poti do drugih storitev in poti, ki jih pripeljejo do tja. Dodana je pot tudi do ELK Stack za centralizirano beleženje dnevnikov.
13. Večnajemništvo
    Na splošno je celotna aplikacija tako zasnovana, da je možno iskanje in dodajanje nepremičnin neodvisno en od drugega, kot je to razvidno na sliki. Ampak tukaj ni bilo implementiranih nekaj posebnih prijemov, za delovanje večnajemništva.



Zaključek

Z izvedbo projekta sem zadovoljen, saj sem se res veliko naučil o različnih delih razvoja oblačnih storitev. Med razvojem sem spremenil nekaj planov saj sem spremenil delitev mikrostoritev. Dodal sem posebno mikrostoritev za slike in posebno za upravljanje z nepremičninami saj se mi je zdela to bolj realna delitev. Spremenil sem tudi uporabniški vmesnik, ki je bil na začetku planiran da bo mobilen (flutter), ampak je bilo bolj praktično integrirati frontend v svojo storitev in povezati preko istega IP-naslova na »ingress« kontrolerju.