Slika, ki vsebuje besede steklenica, oblikovanje, cerkev

Opis je samodejno ustvarjen

Projekt

**ThreatInsight**

**NAČRT**

**PROGRAMSKE OPREME**

Naredil:

**Matevž Nerat**

April 25, 2024

Vsebina

[**1.** **Namen** 3](#_Toc168244697)

[**2.** **Načrt vzpostavitve** 4](#_Toc168244698)

[**2.1.** **Git repozitorij** 4](#_Toc168244699)

[**2.2.** **Jira** 4](#_Toc168244700)

[**2.3.** **SonarCloud** 5](#_Toc168244701)

[**2.4.** **Pročelje** 5](#_Toc168244702)

[**2.5.** **Zaledje** 6](#_Toc168244703)

[**2.6.** **Render** 6](#_Toc168244704)

[**2.7.** **Github Actions** 7](#_Toc168244705)

[**2.8.** **Firebase** 7](#_Toc168244706)

[**3.** **Načrt implementacije** 8](#_Toc168244707)

[**3.1.** **Zaledje** 8](#_Toc168244708)

[**3.1.1.** **Tools** 8](#_Toc168244709)

[**3.1.2.** **Search** 8](#_Toc168244710)

[**3.1.3.** **Authentication** 9](#_Toc168244711)

[**3.1.4.** **Routes** 9](#_Toc168244712)

[**3.1.5.** **Tests** 9](#_Toc168244713)

[**3.1.6.** **Firebase** 10](#_Toc168244714)

[**3.1.7.** **Server** 10](#_Toc168244715)

[**3.2.** **Pročelje** 10](#_Toc168244716)

[**3.2.1.** **Firebase** 10](#_Toc168244717)

[**3.2.2.** **Pages** 10](#_Toc168244718)

[**3.2.3.** **Components** 11](#_Toc168244719)

[**3.2.4.** **Context** 11](#_Toc168244720)

[**3.3.** **Komunikacija med zaledjem in pročeljem** 12](#_Toc168244721)

[**4.** **Načrt podatkovne baze** 12](#_Toc168244722)

[**5.** **Načrt testiranja** 12](#_Toc168244723)

[**5.1.** **Testiranje enot** 12](#_Toc168244724)

[**5.2.** **Funkcionalno testiranje** 13](#_Toc168244725)

# **Namen**

Namen načrta programske opreme je zagotavljanje načina komunikacije z razvojno ekipo in product owner-jem. Razvojna ekipa se je dolžna držati splošnih navodil tega dokumenta, ki okvirno predstavi zaporedje aktivnosti pri razvoju in način njihove implementacije.

Cilj dokumenta je predstaviti neko ogrodje informacijske rešitve, ki je jasna vsem sodelujočim. Opisani so problemi in načini njihovih rešitev. Opisana je razdelitev projektne ekipe ter vloge pri vsaki aktivnosti.

# **Načrt vzpostavitve**

## **Git repozitorij**

Odgovoren: Matevž Nerat

Opis:

Opisana vzpostavitev in konfiguracija Git repozitorija za projekt, ki bo gostovan na GitHubu. Cilj je zagotoviti centralizirano mesto za shranjevanje izvorne kode in dokumentacije, kjer lahko vsi člani ekipe enostavno sodelujejo.

Aktivnosti:

* Vzpostavitev git repozitorija na GitHub
* Dodajanje članov projektne ekipe
* Dodajanje mentorja/product owner-ja
* Nastavljanje settings-ov
* Definicija strukture git repozitorija (frontend, backend, Dokumentacija(vodstveni, specialistični, predloge in vmesni dokumenti))

## **Jira**

Odgovoren: Teo Verhovnik

Opis:

Vzpostavitev in konfiguracija Jira projekta, ki bo uporabljen za spremljanje napredka in organizacijo dela v ekipi po SCRUM metodi. Namen je vzpostaviti učinkovito okolje za upravljanje nalog in zagotoviti preglednost nad stanjem projekta.

Aktivnosti:

* Ustvarjanje novega projekta
* Dodajanje članov projektne ekipe
* Ustvarjanje User Story-jev po vzgledu dokumenta Vizija in SZPO
* Ustvarjanje Scrum strukture (board, backlog, sprinti,…)

## **SonarCloud**

Odgovoren: Matevž Nerat

Opis:

Vzpostavitev SonarCloud projekta, ki bo uporabljen za zagotavljanje kakovosti in varnosti kode, ter kontinuirano spremljanje tehničnega dolga. Namen je vzpostaviti sistem za avtomatizirano analizo kode, ki bo pomagal ekipi pri vzdrževanju visokih standardov kakovosti.

Aktivnosti:

* Ustvarjanje SonarCloud projekta
* Dodajanje članov projektne ekipe
* Nastavljanje Quality Gate-a
* Generiranje potrebnih stvari za Github Actions integracijo

## **Pročelje**

Odgovoren: Matevž Nerat

Opis:

Vzpostavitev osnovne strukture front-end aplikacije z uporabo React in TypeScript. Namen je ustvariti temelje za razvoj uporabniškega vmesnika, ki vključuje osnovne potrebne odvisnosti.

Aktivnosti:

* Ustvarjanje React aplikacije v root-u repozitorija z »npx create-react-app frontend --template typescript«
* Dodajanje potrebnih osnovnih odvisnosti (react-router, bootstrap, axios, jest, firebase)
* Ustvarjanje osnovne strukture (Pages, Components, Context, Firebase)

## **Zaledje**

Odgovoren: Matevž Nerat

Opis:

Vzpostavitev osnovne strukture zaledja (backend) z uporabo Express.js in drugih osnovnih odvisnosti. Namen je zagotoviti osnovo za razvoj strežniškega dela aplikacije, ki bo podpiral vse potrebne funkcionalnosti.

Aktivnosti:

* Ustvarjanje direktorija »backend« v root-u repozitorija
* Ustvarjanje npm projekta
* Dodajanje Express.js ogrodja
* Dodajanje potrebnih osnovnih odvisnosti (body-parser, axios, cors, nodemon, firebase-admin, dotenv)
* Ustvarjanje osnovne strukture (routes, tools, search, authentication, tests)

## **Render**

Odgovoren: Teo Verhovnik

Opis:

Vzpostavitev in konfiguracija hosting okolja za tako statične kot dinamične dele aplikacije z uporabo platforme Render. Namen je omogočiti neprekinjeno dostavo (CD) in zagotavljanje zanesljivega gostovanja za front-end in back-end projekta.

Aktivnosti:

* Ustvarjanje Static site hosting strani
* Povezovanje /frontend git poddirektorija za CD
* Nastavljanje nmp build parametrov
* Ustvarjanje Web service hosting strani
* Povezovanje /backend hit poddirektorija za CD
* Ustvarjanje zagonskih parametrov

## **Github Actions**

Odgovoren: Matevž Nerat

Opis:

Vzpostavitev avtomatiziranih CI procesov z uporabo Github Actions, ki zagotavljajo avtomatsko testiranje in analizo kode ob vsakem potisku sprememb v repozitorij. Namen je zagotoviti visoko kakovost kode in hitro povratno informacijo za razvojno ekipo.

Aktivnosti (Node.js CI):

* Ustvarjanje workflow-a »Node.js CI« 🡪 node.js.yml.
* Uporaba template-a »Node.js«
* Konfiguracija node.js.yml
  + Nastavljanje proženja na »push«
  + Nastavljanje verzije (18.x)
  + Nastavljanje korakov (actions/checkout@v4, actions/setup-node@v3, npm ci, npm test, ustvarjanje skrivnosti)
  + Nastavljanje working directory: ./backend

Aktivnosti (SonarCloudAnalysis):

* Ustvarjanje workflow-a »SonarCloudAnalysis« 🡪 sonarcloud-CI.yml
* Konfiguracija sonarcloud-CI.yml
  + Nastavljanje proženja na »push«
  + Nastavljanje JDK 17
  + Nastavljanje SonarSource/sonarcloud-github-action@master
  + Uporaba github skrivnosti
  + Argumenti do SonarCloud projekta

## **Firebase**

Odgovoren: Teodor Verhovnik

Opis:

Integracija Firebase v projekt, vključno z vzpostavitvijo projektne infrastrukture in konfiguracijo storitev, kot so Firebase Authentication in Firestore.

Aktivnosti:

* Ustvarjanje projekta
* Dodajanje članov projektne ekipe
* Vzpostavitev Firebase Authentication
* Vzpostavitev Firestore
  + Osnovna struktura kolekcije in dokumentov(users/id/search/id)

# **Načrt implementacije**

## **Zaledje**

Vsako podpoglavje predstavlja del strukture programske kode v zaledju.

### **Tools**

Odgovorni:

* Matevž Nerat (nmap, Domain Search, TLS-DNSSec)
* Aljaž Bračko (Permutator, hibp, WhoIs)
* TeodorVerhovnik (ipGeo)

Direktorij »tools« je namenjem pridobivanju podatkov vsakega orodja/storitve posebej. Vsebuje 7 pod direktorijev, vsak za svoje orodje. Pod direktoriji so poimenovani po imenu orodja oz. njihovi krajšavi(npr. HaveIBeenPwned 🡪 hibp). Vsi pod direktoriji orodij vsebujejo vsaj 5 datotek, ki so:

* **[imeOrodja].js**: glavna datoteka orodja, ki je klicana, ko se uporabnik ustvari iskanje z tem orodjem. Funkcija v datoteki je istoimenska, kot datoteka. Prejme podatke potrebne za orodje, userUID in searchUID. V primeru uspeha vrne objekt, kjer je uspeh potrjen. V primeru neuspeha vrne objek, kjer je uspeh zavrnjen.
* **[imeOrodja]API.js**: datoteka v kateri se implementirajo vsi potrebni klici na API-je storitev
* **[imeOrodja]StructureResponse.js**: prejete odgovore API-jev preoblikuje v strukturo, ki jo potrebujemo
* **[imeOrodja]SaveResponse.js**: shrani preoblikovano strukturo odgovora orodja v podatkovno bazo
* **[imeOrodja].test.js**: datoteka z testi za vse datoteke orodja
* + druge datoteke po potrebi integracije in preverjanje pravilnosti

### **Search**

Odgovoren: Matevž Nerat

Direktorij »search« ima 3 pod direktorije:

* **createSearch:**
  + poskrbi za vse od ustvarjanja iskanja, klicanja vseh orodij, pridobivanja rezultatov orodij, shranjevanja vmesnih rezultatov orodij v podatkovno bazo, čakanje na končanje izvajanja vsakega orodja, pridobivanje vmesnih rezultatov iz baze, preoblikovanje vmesnih rezultatov v končni rezultat, shranjevanje končnega rezultata v bazo in pošiljanje url poti na frontend za preusmeritev.
* **getSearch:**
  + datoteka getSearch.js, ki predpisuje funkcijo za pridobivanje končnega rezultata iskanja iz podatkovne baze. Prejme parametre userUID, searchUID.
* **updateSearch**
  + datoteka updateSearch.js, ki predpisuje funkcijo za posodabljanje imena in opisa iskanja. Prejme parametre userUID, searchUID, name in notes.

### **Authentication**

Odgovoren: Teodor Verhovnik

Direktorij namenjen vsem potrebnim funkcijam za avtentikacijo.

### **Routes**

Odgovorna:

* Matevž Nerat (search)
* Teodor Verhovnik (auth)

Direktorij »routes« je direktorij, ki poskrbi za komunikacijo med pročeljem in zaledjem. Vsebuje endpointe, ki ji ponuja zaledje. Za varnost je implementirana middleware funkcija, ki preveri JWT token in njegovo veljavnost. Ima dva pod direktorija: auth in search. Auth je namenjem vsemu povezanemu z avtentikacijo.

Search ima 3 endpointe, vsi prožijo ustrezne funkcije iz direktorija »search«:

* POST /search/
  + Klice funkcijo v datoteki createSearch.js, ki zažene proces iskanja.
* GET /search/
  + Pridobi končni rezultat iskanja.
* PUT /search/
  + Posodobi ime in opis iskanja.

### **Tests**

Odgovoren: Matevž Nerat

Direktorij namenjen testom, ki niso neposredno povezani z orodji.

### **Firebase**

Odgovoren: Matevž Nerat

Datoteka firebase.js ponuja dve funkciji: initializeFirestore in getFirestoreInstance.

InitializeFirestore inicializira instanco za dostop do funkcij Firestora.

GetFirestoreInstance vrača to instanco.

Implementirano po vzorcu edinec (singleton).

### **Server**

Odgovorna:

* Matevž Nerat
* Teodor Verhovnik

Začetna datoteka server.js, ki inicializira celotno backend express aplikacijo. Med drugim proži funkcijo initializeFirestore in poveže direktorij »routes« z express aplikacijo.

Aplikaciji doda CORS preverjanje in JSON body parser.

## **Pročelje**

### **Firebase**

Odgovoren: Teodor Verhovnik

Direktorij Firebase in datoteka firebase.tsx poskrbi za uvoz in izvoz vseh potrebnih funkcij Firebase SDK-ja.

### **Pages**

Odgovorna:

* Aljaž Bračko (HomePage, ScansPage – prikaz rezultatov iskanja, NewScanPage)
* Teodor Verhovnik ( DocsPage - delovanje, AuthenticationPage, ProfilePage, YourWorkPage)
* Matevž Nerat (DocsPage – vsebina)

Direktorij, ki vsebuje pod direktorije vsake strani pročelja. Vsak direktorij strani ima smiselno razdeljene pod direktorije potrebnih komponent tiste strani:

* HomePage
* DocsPage
* ScansPage
* AuthenticationPage
* NewScanPage/SearchPage
* ProfilePage
* YourWorkPage

Načrtovani izgledi vseh strani so na voljo v direktoriju /Dokumentacija/Predloge in vmesni dokumenti/Wireframes.

### **Components**

Odgovorna:

* Aljaž Bračko (Header, Footer)
* Teodor Verhovnik (ScansHeader – side menu, LoadingOverlay, VerticalHeader)
* Matevž Nerat (ScansHeader – edit modal)

Direktorij namenjen skupnim komponentam celotne strani. Tukaj se dodajo Header-ji, Footer, loading screen-i,…

### **Context**

Odgovoren: Teodor Verhovnik

Namenjen ustvarjanju Contextov.

Prvi Context je AuthContext, namenjen vzdrževanju stanja prijave in prijavi.

Drugi Context je FirebaseErrorsContext, namenjen preoblikovanju firebase error-jev v berljivo podobo.

## **Komunikacija med zaledjem in pročeljem**

Odgovorni:

* Aljaž Bračko
* Teodor Verhovnik
* Matevž Nerat

Zaledne funkcionalnosti so ponujene preko REST API-ja, ki ni namenjen javni uporabi, ampak samo za interno delovanje aplikacije.

Za ustvarjanje klicev na pročelju je uporabljen Axios.

Implementacija na način opisan v poglavju 3.1.4. Routes.

# **Načrt podatkovne baze**

Izbrana podatkovna baza je Firestore. Sama po sebi je NoSQL dokumentno orientirana podatkovna baza, ki deluje na principu kolekcij njim ustreznim dokumentom.

Za potrebe aplikacije je ustvarjena osnovna struktura:

* Vrhovna kolekcij »users«, ki vsebuje dokumente poimenovane z uporabniškimi id-ji povezanimi iz Firebase Authentication storitve.
* Vsak users dokument ima kolekcijo »iskanje«, ki vsebuje dokumente poimenovane z naključno ustvarjenim id-jem.
* Vsak »iskanje« dokument ima: name, notes, creation date, objekte poimenovane po izbranih orodjih za tisto iskanje, ter končni rezultat, ki vsebuje preoblikovan rezultat vseh orodij v iskanju.

# **Načrt testiranja**

## **Testiranje enot**

Za testiranje enot je izbrano ogrodje Jest, ki je namenjeno testiranju JavaScript projektov.

Cilj testiranja enote je čim večja pokritost enot z testi. Teste implementira razvijalec sam po implementaciji enote. Ob padcu kateregakoli testa popravi novo kodo na način, ki ne vpliva na druge komponente aplikacije.

Testi se zaženejo s pomočjo node in sicer z: npm test

## **Funkcionalno testiranje**

Za pravilno delovanje aplikacije so napisani testni scenariji (**ThreatInsight-Dokumentacija\_testiranja.docx),** s katerimi preverjamo funkcionalne zahteve iz SZPO.