Mobilne i web-aplikacije za Internet stvari s primjenom u poljoprivredi

Tehnička dokumentacija

Verzija 1.0

Studentski tim: Marko Brlek

Filip Fabris

Josipa Jović  
Karla Kijac  
Lovro Makovec  
Fran Marić  
Luka Radman  
Valentina Valić  
Ivan Žgela

Nastavnici: Prof. dr. sc. Gordan Ježić

Prof. dr. sc. Mario Kušek

Sadržaj

[**1. Opis razvijenog proizvoda 3**](#_Toc125132336)

[1.1 Korištene tehnologije 3](#_Toc125132337)

[1.2 Elementi proizvoda 4](#_Toc125132338)

[1.2.1 Poslužitelj 4](#_Toc125132339)

[1.2.2 Flutter aplikacija 4](#_Toc125132340)

[1.2.3 Web-aplikacija 4](#_Toc125132341)

[1.2.4 Mikroupravljač 4](#_Toc125132342)

[**2. Tehničke značajke 5**](#_Toc125132343)

[2.1 Opis arhitekture 5](#_Toc125132344)

[2.2 Detaljniji opis arhitekture 5](#_Toc125132345)

[2.2.1 Poslužitelj 5](#_Toc125132346)

[2.2.2 Arhitektura Web aplikacije 9](#_Toc125132347)

[2.2.3 Arhitektura Flutter aplikacije 11](#_Toc125132348)

[**3. Upute za korištenje 12**](#_Toc125132349)

[3.1 Poslužitelj 12](#_Toc125132350)

[3.2 Flutter aplikacija 13](#_Toc125132351)

[3.3 Web aplikacija 15](#_Toc125132352)

[3.4 Mikroupravljač, senzori, LoRaWAN Gateway i ChirpStack poslužitelj 30](#_Toc125132353)

[3.4.1 Mikroupravljač i senzori 30](#_Toc125132354)

[3.4.2 LoRaWAN Gateway i ChirpStack poslužitelj 35](#_Toc125132355)

[**4. Literatura 38**](#_Toc125132356)

Tehnička dokumentacija

# Opis razvijenog proizvoda

U sklopu projekta bilo je potrebno izraditi REST poslužitelj koji u svoju bazu podataka sprema informacije o scenama, prikazima i ključevima za pristup podatcima baze podataka InfluxDB.

Omogućeno je brisanje i dodavanje scena korištenjem web-aplikacije te pristup REST poslužitelju temeljem OAuth2 autentifikacije preko poslužitelja Keycloak.

Također, mobilna aplikacija u Flutteru koristit će isti REST poslužitelj, ali će samo čitati podatke i prikazivati ih korisniku. U mobilnoj aplikaciji bit će prikazane scene te pripadajući grafovi s obzirom na ulogu korisnika.

Namjena web-aplikacije je upravljanje scenama, kreiranje novih scena, uređivanje postojećih te brisanje određenih scena. Naime, svaka scena ima 2 tipa view-a (actuation view i measurement view). Za svaku scenu je omogućeno kreiranje novih view-a, njihovo uređivanje i brisanje. Podatci postojećih scena i view-a u bazi podataka služe mobilnoj Flutter aplikaciji za prikaz relevantnih podataka iz baze podataka, njihove detalje te prikaz podataka u obliku grafova.

Što se tiče mikrokontrolera (ili mikroupravljača), koji upravlja komunikacijom između senzora i korisničke aplikacije, izvedeno je očitavanje senzora te slanje očitanja i poruka na LoRaWAN Gateway. Mikrokontroler je spojen na senzor razine vode.

## Korištene tehnologije

Za izradu REST poslužitelja korištena je Java uz radni okvir Spring te relacijska baza podataka PostgreSQL.

Za autentifikaciju, autorizaciju i upravljanje identitetom korišten je Keycloak poslužitelj koji omogućava implementiranje različitih tipova provjere identiteta (autentifikacije) i provjere ovlaštenja (autorizacije), namijenjen modernim aplikacijama i uslugama. [1]

Flutter se koristio za izradu mobilne aplikacije, a okruženja u kojima je aplikacija razvijena su Visual Studio Code te Android Studio. Za izradu grafova koristio se package iz Flutter Charts knjižnice (eng. library), a za autentifikaciju package iz Oauth2 knjižnice. Osim toga korišteni su Xcode Simulator te Android Emulator.

Razvojno okruženje u kojem je razvijena web-stranica je Visual studio code. Web-stranica izrađena je pomoću React radnog okvira. Prilikom izrade web stranice je korišten jezik TypeScript (inačica javascripta koja ima definirane tipove podataka), Axios biblioteka (eng. library) za slanje HTTP zahtjeva na backend server, Keycloak biblioteka za autorizaciju korisnika te mnoge druge knjižnice.

Komunikacija na daljinu u timu je ostvarena korištenjem aplikacija Discord video poziva i GoogleMeet-a.

Za programiranje mikrokontrolera STM32F407VGT6 (ARM Cortex-M4), smještenog na Clicker 2 razvojnoj pločici, korišten je NECTO Studio koji je kao razvojno okruženje ponudila Mikroelektronika čiji je i Clicker 2. Za spajanje na *gateway* koristi se još PuTTY koji omogućava komunikaciju između računala putem terminala i protokola SSH (najčešće). Koriste se dva poslužitelja od ChirpStacka, LoRaWAN Network Server i LoRaWAN Application Server. Mrežne prilaze (eng. *gateway*) možemo konfigurirati, dodavati i vidjeti spojene uređaje, te poruke koje dolaze s uređaja. Uređaji se također mogu, i trebaju dodati kako bi se mogli aktivirati i spojiti na gateway.

## Elementi proizvoda

### Poslužitelj

Osnovna namjena poslužitelja (web-aplikacije) je komunikacija s bazom podataka, u suradnji s web-stranicom. U daljnjem tekstu zbog jednostavnosti, za aplikaciju napisanu u Javi, koristit će se izraz *poslužitelj*, a Tomcat poslužitelj na kojem se nalazi web aplikacija prešutno će se pretpostavljati. Baza podataka PostgreSQL sadrži podatke o scenama, ključevima, ulogama i sl., što služi mobilnoj aplikaciji za prikaz grafova, a ti podaci mogu se mijenjati putem web-stranice. Svakom korisniku dodijeljene su ovlasti pomoću servisa Keycloak, a za svaki upit prema poslužitelju potrebno je imati odgovarajuće ključeve.

### Flutter aplikacija

Mobilna aplikacija, kompatibilna sa iOS i Android uređajima, služi za praćenje sustava. Prijavljenom korisniku omogućava pretraživanje scena, pregled grafova za odabranu scenu po različitim vremenskim periodima za izabrana mjerenja.

### Web-aplikacija

Putem web-aplikacije, na početnoj stranici, mogu se vidjeti opće informacije o projektu i IoT polju. Stranica nudi detaljan pregled scena i svih prikaza povezanih sa određenom scenom. Osim prikazivanja scena, one se mogu dodavati, uređivati i brisati.

### Mikroupravljač

Mikroupravljač, Clicker 2, upravlja senzorima, pumpom i ventilima za navodnjavanje. Mjerenja koja prima od senzora šalju na LoRaWAN Gateway. Podaci su kodirani pa ih je potrebno i dekodirati.

# Tehničke značajke

## Opis arhitekture

Sustav se sastoji od četiri dijela: Mobilne aplikacije, Web-aplikacije, baze podataka i mikroupravljača sa senzorima u vrtu.

* **Mobilna aplikacija –** služi za pristup grafovima i podacima sa senzora. Komunicira s Keycloakom za *login* korisnika i REST poslužiteljem za dohvaćanje scena i podataka.

Web aplikaciju možemo podijeliti na:

* **Web-preglednik -** je program koji služi za pristup web stranicama. Putem web preglednika, korisnik šalje zahtjeve za resursima (npr. HTML kod web-stranice) ili šalje podatke (npr. putem neke forme), web-preglednik dohvaća te datoteke s web poslužitelja, a potom ih interpretira i prikazuje na ekranu korisnika ili ih pohranjuje na poslužitelju.
* **Web-poslužitelj -** glavni je dio web-aplikacije. To je namjensko računalo ili software koji šalje i prima podatke od mnogostrukih klijenata. Komunikacija s klijentima (korisnicima i bazom podataka) odvija se preko HTTP protokola. Na korisnikov zahtjev, web preglednik dohvaća resurse i vraća u obliku HTML dokumenta ili obraduje podatke predane u formi te ih sprema u bazu podataka.
* **Mikroupravljač, senzori i LoRaWAN Gateway –** zaslužni su za izvedbu navodnjavanja u vrtu i slanje/primanje poruka/naredbi.
* **Baza podataka -** koristi se za pohranjivanje podataka sustava. Web-aplikacija u svom radu vrlo često komunicira s bazom te iz nje dohvaća podate ili ih u nju sprema

Pri oblikovanju aplikacije koristili smo MVC (Model-View-Controller) obrazac softverske arhitekture. Po principu MVC-a, aplikaciju dijelimo na tri komponente:

* Model - je glavna komponenta sustava. Predstavlja strukturu podataka (Java objekti) i njihovu funkcionalnost.
* View - odlučuje kako će se dohvaćeni podaci reprezentirati.
* Controller - zaprima zahtjeve za resursima (HTTP zahtjevi) od klijenta koje prilagođava i prosljeđuje Modelu ili Viewu.

## Detaljniji opis arhitekture

### Poslužitelj

Poslužitelj se temelji na REST arhitekturi i Keycloak autentifikaciji. Poslužitelj je napisan pomoću radnog okvira Spring, koji radi tako da iskombinira razrede aplikacije zajedno s konfiguracijskim metapodacima u kontejneru za inverziju upravljanja, nakon čega se dobiva povezani sustav koji se može koristiti.

Spring koristi arhitekturni obrazac MVC: *model*, *view* i *controller*, tj. model, pogled i nadglednik. Model je ostvaren slojevima *Service* i *Repository* te on prima zahtjeve od nadglednika. Poslužitelj koristi više razreda, npr. *Scene*, *Tag*, *View*, *Role*, *Layout*, *Key*, *Inputs*…

Diagram

Description automatically generated

*Slika 2.2.1.1 Dijagrami nekih razreda*

Dakle, Spring ima 3 sloja koja međusobno komuniciraju i najčešće se ne preskaču: klijentska strana komunicira s *Controllerom*, a on sa *Service* i *Repository*, dok baza podataka komunicira s *Repositoryjem*. S druge strane, DTO (*Data Transfer Object*) su objekti namijenjeni za prijenos podataka. U sloju *Controller*, komponente se označavaju anotacijama (npr. *@RequestMapping*, *@Controller* i sl.) Isječak programskog kȏda u nastavku prikazuje primjer funkcionalnosti sloja *Controller*.

|  |
| --- |
| **SceneController.java** |
| @GetMapping("/scene") public ResponseEntity<List<ShortSceneDTO>> getScenes() {  List<Scene> list = service.getAllScenes();  List<ShortSceneDTO> shortScenes = new ArrayList<>();   for (Scene scene : list) {  shortScenes.add(ShortSceneDTO.*of*(scene));  }  return ResponseEntity.*status*(HttpStatus.*OK*).body(shortScenes); }  @RolesAllowed("iot-read") @GetMapping("/scene2") public ResponseEntity<List<ShortSceneDTO>> getScenes2() {   List<Scene> list = service.getAllScenesAuthorize2();  List<ShortSceneDTO> shortScenes = new ArrayList<>();  for (Scene scene : list) {  shortScenes.add(ShortSceneDTO.*of*(scene));  }  return ResponseEntity.*status*(HttpStatus.*OK*).body(shortScenes); } |

Vidljivo je da navedeni *Controller* odgovara na GET zahtjev */scene* izvršavajući funkciju *getScenes()*. Navedena funkcija vraća HTTP OK status zajedno s popisom scena. Funkciju *getScenes2()* mogu izvršavati samo korisnici koji imaju *iot-read* ulogu, a dohvaća se putem GET zahtjeva */scene2*. Anotacije dakle služe Springu za ostvarenje sloja *Controller*.

Što se tiče načina rada REST API-a, on se zasniva na temeljnim HTTP metodama: *GET*, *POST*, *PUT*, *PATCH* i *DELETE*. Najčešće, *GET* služi za siguran dohvat podataka, *POST* za stvaranje novih podataka, *PUT* za uređivanje, a *DELETE* za brisanje. Te metode implementiraju se u *Controller* sloju pomoću anotacija, kao u primjeru iznad.

Zadaća sloja *Service* je ostvarivanje temeljne funkcionalnosti aplikacije, u smislu da definira koje parametre prima, što se treba ostvariti te upravlja transakcijama prema bazi podataka. U primjeru ove aplikacije, korištena su tri sloja: *KeyService*, *SceneService* i *TagService*. Na primjer, autorizacija pomoću Keycloaka obavlja se u sloju *Service*:

|  |
| --- |
| **SceneService.java** |
| public Scene AddSceneAuthorize(Scene scene) {    HashSet<String> rolesKeyCloak = new HashSet<>(KeycloakSecurityConfig.*getRoles*().stream().map(role -> role.toString().split("\_")[1]).toList());   if (rolesKeyCloak.size() < 1) {  throw new NoSuchElement("Not authorized");   }  if (!sceneRepository.existsById(scene.getId())) {  return sceneRepository.save(scene);  }  throw new NoSuchElement("Scene " + scene.getId() + " already exists!");  } |

Budući da su uloge spremljene u Keycloak te da bi se mogla izvršiti autorizacija korisnika i njihovih uloga, navedenim isječkom kȏda dohvaćaju se uloge te se koristeći *map()* funkciju spremaju u listu. Funkcija *AddSceneAuthorize* sprema scene u bazu podataka koristeći sloj *Repository*, provjeravajući ispravnost uloga.

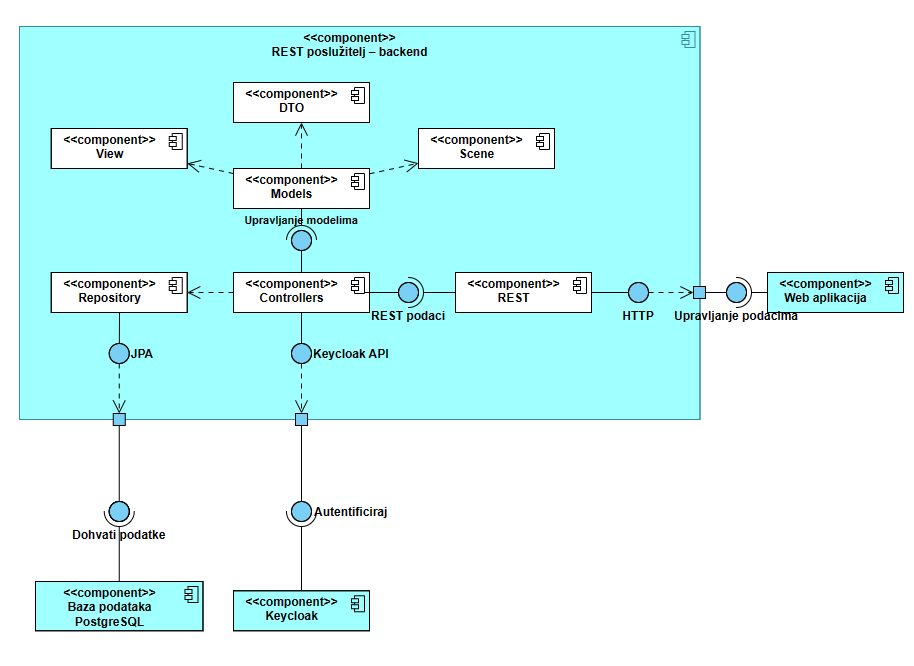
Komunikacija s bazom podataka radi preko sloja *Repository*, a unutar toga su ostvarene i usluge JDBC i JPA. Korištena baza podataka je relacijska PostgreSQL te su u aplikaciji korišteni slični razredi kao i za sloj *Controller*: *SceneRepository* i *TagRepository*. Iako se naredbe u tim razredima mogu napisati bez njih, korištenjem sloja *Repository*, teret dohvata podataka pada na bazu podataka.

|  |
| --- |
| **SceneRepository.java** |
| @Query(value="SELECT scene.\* FROM ROLE INNER JOIN scene ON role.scene\_id = scene.id WHERE role.name IN :roles", nativeQuery = true) Collection<Scene> dobijSceneSOvimRolama(@Param("roles") Collection<String> roles);  @Query(value="SELECT scene.\* FROM ROLE INNER JOIN scene ON role.scene\_id = scene.id WHERE role.name IN ?1 AND scene.id = ?2", nativeQuery = true) Scene dobijSceneSOvimRolamaiID(Collection<String> roles, Long findID); |

I ovdje se može uočiti korištenje anotacija u kojima se nalazi SQL koji želimo da se izvrši. Kao što je prethodno rečeno, dohvat scena s određenim ulogama mogli bismo napisati u sloju *Controller* ili *Service*, no tada bi se zauzimali resursi poslužitelja. Na ovaj način, sortiranje scena obavlja se na bazi podataka, odvojeno od poslužitelja.

**Dijagram komponenti**

Dijagram komponenti na slici 2.2.1.2. prikazuje međuovisnost, organizaciju i odnose u internoj strukturi. Sustavu se pristupa preko internetskog preglednika preko sučelja za dohvat HTML, CSS i JS datoteke koje služe za prikaz i funkcionalnosti grafičkog sučelja. Router je komponenta kojom se upravlja prikaz Internet stranica. Sučelje za primanje JSON podataka pristupa se preko REST API komponenti. Pomoću Axios biblioteke dobivene podatke s backenda prenosimo na korisnikov preglednik. Na backendu nam je potrebna komponenta Controller koja služi da modele koje pretvaramo u DTO objekt pošaljemo na frontend.



*Slika 2.2.1.2.* Dijagram komponenti za poslužitelj

### Arhitektura Web aplikacije

Opcije koje nudi aplikacija su pregled SCENA, KLJUČEVA i VIEW-OVA.

**SCENE -** Prilikom odabira opcije pregleda scena prikazuje se tablica sa popisom scena. Korisnik ima opciju i pretraživanja tablice kako bi na jednostavan način filtrirao tablicu i pronašao naziv scene koju želi pregledati, urediti ili obrisati. Unutar tablice u zadnja 2 stupca su opcije za uređivanje i brisanje posebne scene. Prilikom odabira „uredi“ otvara se popunjena forma s relevantnim podatcima koje korisnik može urediti. Odabirom opcije obriši korisnik mora potvrditi brisanje iz sigurnosnih razloga (potencijalno se briše velik broj podataka za scenu i cijeli popis view-a koji pripadaju toj sceni). Kako bi se prikazali detalji pojedine scene i njeni view-i korisnik mora odabrati scenu u tablici. Klikom na redak tablice otvaraju se pojedinosti odabrane scene. Na tom prikazu korisnik također ima mogućnost uređivanja i brisanja scene. Ispod slike scene nalazi se tablica s popisom view-a (actuation i measurement view-i). Također, view-ovi se mogu filtrirati tako da se unese u tražilicu nazivi željenog view-a.

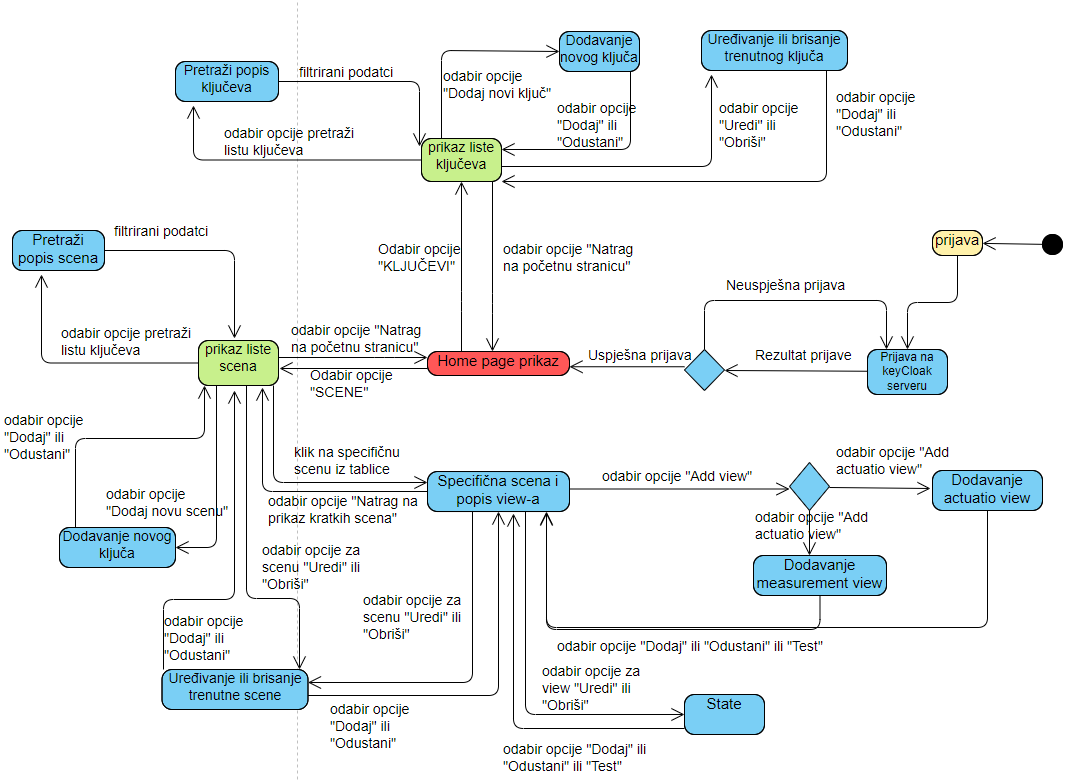
**VIEW –** Postoje dvije opcije za dodavanje view-a. Ovisno o korisnikovim zahtjevima korisnik može dodati actuation ili measurement view. Odabirom jedne od opcija se otvara forma u kojoj korisnik unosi podatke za kreiranje novog view-a. Prilikom kreiranja novog view-a korisnik ima mogućnost testiranja gdje se može uvjeriti jesu svi uneseni podatci ispravni i ako nisu dobije poruku koji podatci nisu ispravni te na temelju toga može ispraviti svoje pogreške. Kao i kod scena korisnik može urediti ili obrisati željeni view tako da u tablici odabere opciju za uređivanje ili za brisanje.

**KLJUČEVI** - Prilikom odabira opcije pregleda ključeva korisniku se prikaže popis postojećih ključeva u bazi podataka. Korisnik ima u desnom kutu opciju za dodavanje novog ključa. Prilikom dodavanja otvara se forma za dodavanje ključa gdje korisnik ima opciju dodavanja ključa ili da odustane od tih promjena. Unutar tablice se nalaze opcije za uređivanje pojedinog ključa. Prilikom odabira uređivanje ključa otvara se popunjena forma s relevantnim podatcima odabranog ključa i korisnik može izmijeniti samo vrijednost ključa (onemogućena mu je izmjena naziva jer je on jedinstven). Prilikom brisanja korisnik mora potvrditi brisanje kao kod scena i view-a.

U svakom trenutku korisnik može s bilo kojeg prikaza odabrati opciju “POČETNA STRANICA” ili “SCENE” ili “KLJUČEVI” koje se nalaze u navigacijskoj traci na vrhu stranice ili klikom na ikone s lijeve odnosno desne strane na navigacijskoj traci.

**Dijagram stanja**

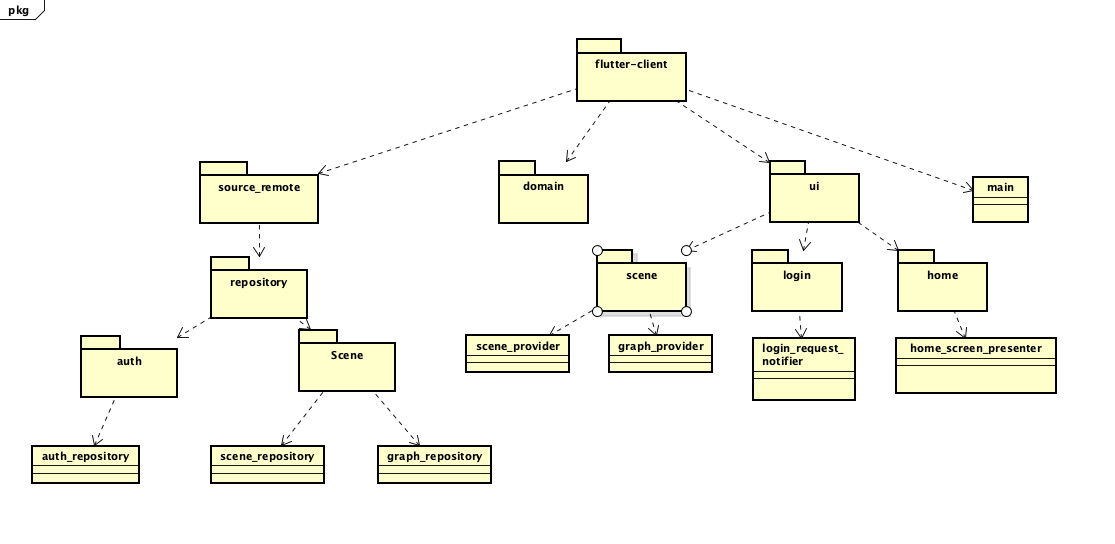
Na slici 2.2.2.2. je prikaz dijagrama stanja koji predstavlja stanja kroz koje korisnik prolaze od trenutka prijave, pa sve do trenutka odjave iz web aplikacije. Početno stanje za korisnika je prijava na Keycloak serveru. Ako je prijava uspješna korisniku je prikazana početna stranica ”Home page” gdje se nalazi animacija za IoT polje i mogućnosti koje nudi ova web stranica.



Slika 2.2.2.1 Dijagram stanja

### Arhitektura Flutter aplikacije

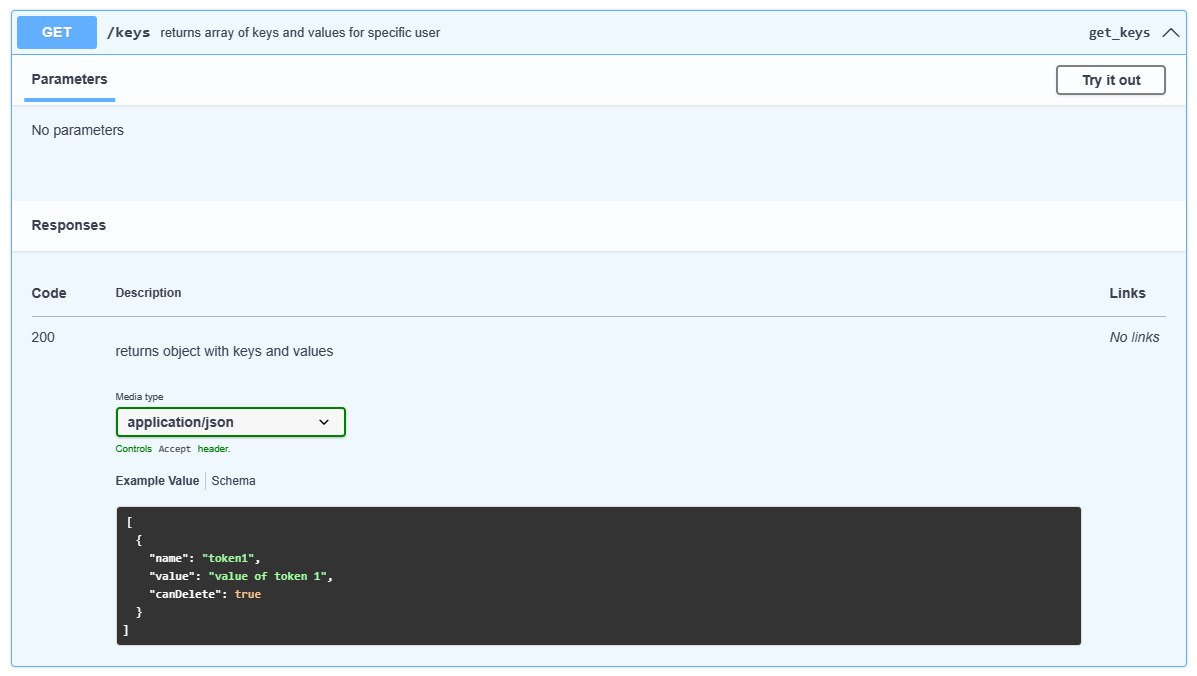
Flutter aplikacija izgrađena je po uzoru na VIPER arhitekturu (slika 2.2). Za svaki modul VIPER ima pet različitih razreda s različitim ulogama. Nijedna klasa ne ide dalje od svog jedinog cilja. Ti razredi su sljedeći *View, Presenter, Interactor, Router, Entity*. Klasa *View* je zadužena za korisničko sučelje aplikacije, klasa *Presenter* dobiva korisnikov odgovor od *Viewa,* klasa Interactor ima poslovnu logiku aplikacije, klasa *Router* sluša *Presenter* klasu o tome koji ekran treba prikazati, klasa *Entity* sadrži model klasa koje koristi *Interactor*. U našoj aplikaciji sve se nalazi u mapi flutter-client koja sadrži *main* klasu kojom pokrećemo aplikaciju te tri mape *source\_remote, ui* i *domain.* Mapa *source\_remote* sadrži mape di, dio i repository. Mapa dio sadrži klase poput *ApiEndpoints* koja sadrži potrebne url-ove za dohvaćanje podataka. Mapa repository sadrži podmape auth i scene unutar kojih nalazimo klase *auth\_repository*, *scene\_repository* i *graph\_repository*. Navedene klase su ključne za autentifikaciju korisnika, dohvaćanje scena i podataka za izradu grafova iz baze podataka. Unutar mape *ui* nalazimo klase potrebne za prikaz različitih ekrana: *home*, *login* i *scene.* Za prikaz scene koristi se scene\_provider, za prikaz grafova unutra scene koristimo graph\_provider. Navedene klase pozivaju metode iz klasa u mapi repository te tako dobivaju dohvaćene podatke. Zatim se scene\_provider i graph\_provider pozivaju unutar odgovarajućih ekrana gdje se njihovi podaci prikazuju na ekranu. Za prikaz početnog ekrana koristi se home\_screen\_presenter, a za prijavu korisnika login\_request\_notifier. Unutar mape *ui* se još nalaze komponente korisničkog sučelja tzv. *Widgets*. Nije prikazano na slici ali arhitektura sadrži još i vanjske medijateke korištene za autentifikaciju i prikazivanje grafova.

*Slika 2.2.3.1. Prikaz nekih od mapa i klasa*

# Upute za korištenje

## Poslužitelj

Poslužitelj koristi REST, odnosno HTTP zahtjeve. U gornjem poglavlju 2.2.1 navedeni su neki primjeri zahtjeva koje poslužitelj prihvaća i kako odgovara.



*Slika 3.3.1. Prikaz iz swaggera*

U ovom primjeru, kada korisnik pošalje GET zahtjev na */rest2/keys* (definirano u sloju *Controller*), odgovor će mu biti kao što je prikazano na slici. Na sličan način definirane su putanje za scene (*/scene*) i oznake (*/tags*), s dodatnim REST zahtjevima (POST za dodavanje novih scena ili ključeva, PUT za ažuriranje i sl.)

*Swagger* sadrži popis putanja i očekivanih rezultata, kao što je prikazano na slici 3.1. Nalaze se i formati (eng. *schemas*) za objekte razreda. Tako na primjer razred *View* može biti ili *MeasurementsView* ili *ActuationView* koji svaki ima svoje posebnosti. Razred *MeasurementsView* ima dodatne atribute: *measurementUnit*, *selectForm*, *query* i *responseExtracting*.



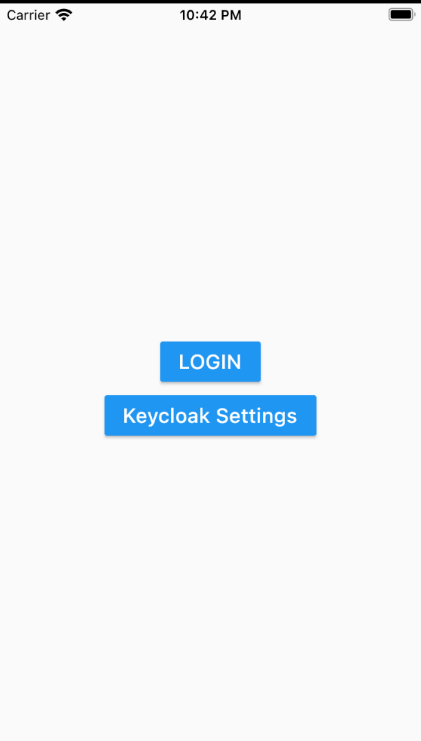
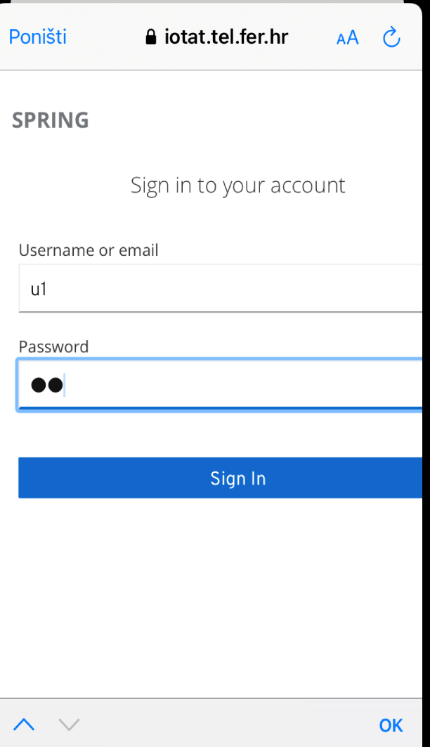
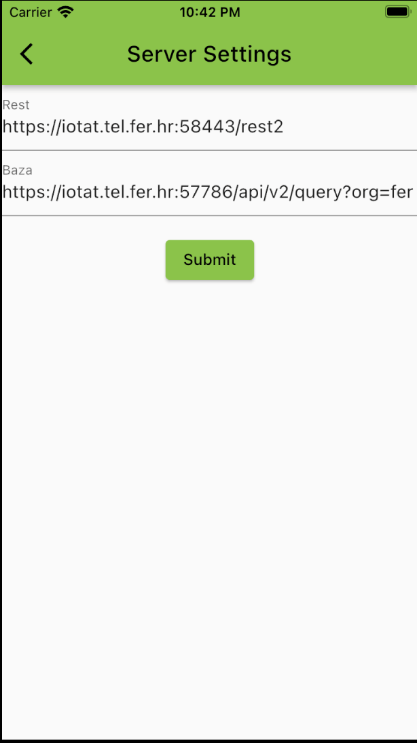
*Slika 3.1.2. Isječak iz swaggera*

Autentifikacija i autorizacija korisnika obavlja se pomoću Keycloaka, u sloju *Controller*. Sustav prepoznaje 4 uloge: *fer*, *ferit*, *iot-read* i *iot-write*. Uloge se dodjeljuju korisnicima (korisnik *u2* ima ulogu *iot-write* a korisnik *u4* uloge *fer* i *iot-read*). Uz uloge su povezane dozvole koje korisnicima pripadaju, npr. korisnik *u4* ne može dodavati nove scene jer nema *iot-write*.

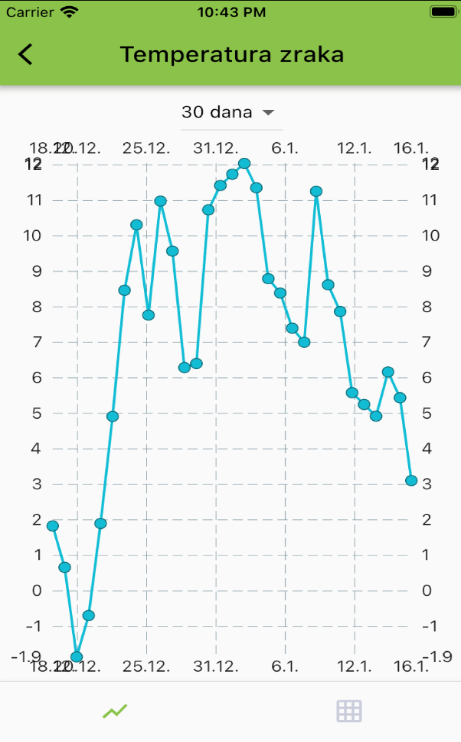
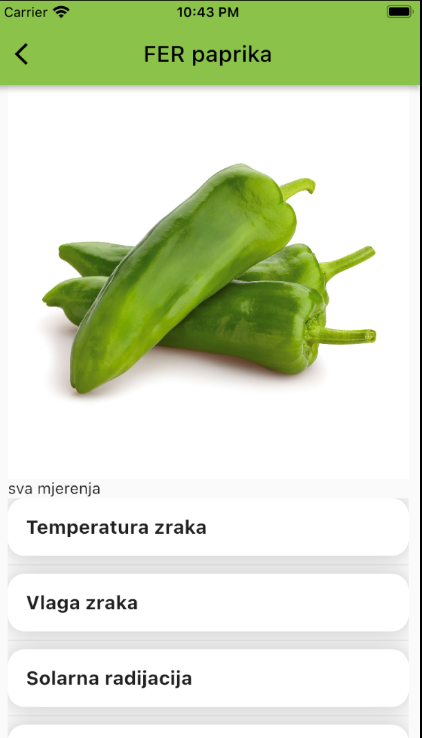
## Flutter aplikacija

Za pokretanje aplikacije preduvjet je imati instalirano Android Studio ili Visual Studio Code. U projektu u main grani potrebno je otići u mapu ' ~/SOFTWARE/flutter\_client/lib' te pokrenuti main.dart pomoću Android Emulatora koji dolazi uz Android Studio ili pomoću Xcode Simulatora.

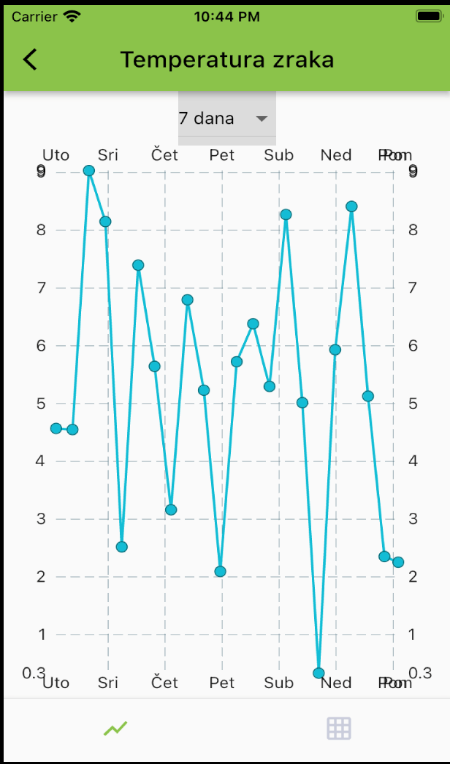
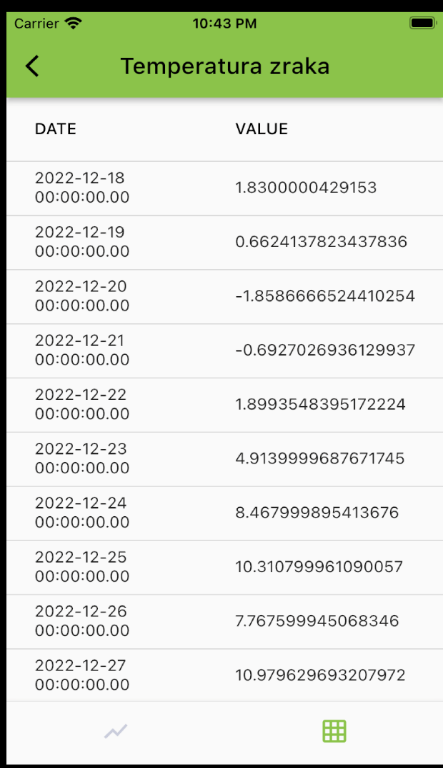
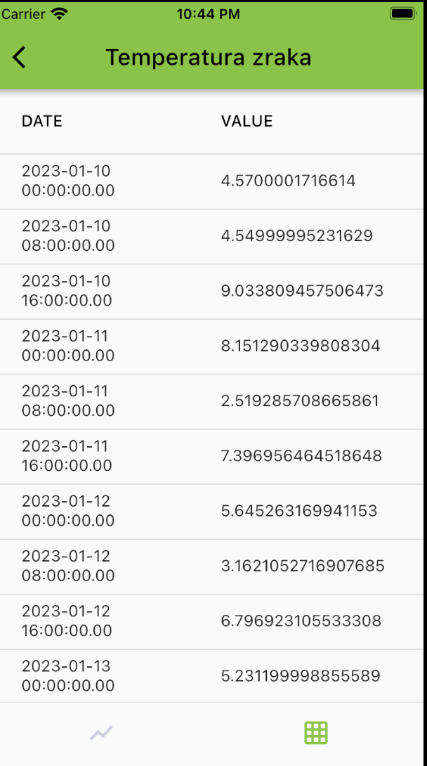
Kada pokrenemo aplikaciju otvara nam se početni zaslon (slika 3.1.) sa gumbima LOGIN i Keycloak Settings. Ako kliknemo na Keycloak Settings otvara nam se ekran (slika 3.2.) u kojemo možemo mijenjati URL REST poslužitelja i URL baze. Ako kliknemo na login aplikacija nas preusmjerava na formu u kojoj je potrebno upisati korisničko ime i lozinku (slika 3.3.). Pritiskom na gumb Sign In šalje se POST zahtjev na Keycloak, a kao odgovor dobivamo *Acces* i *Refresh* token ako su podaci ispravni. U slučaju da podaci nisu bili ispravni dobivamo poruku o neispravnosti podataka. Ako je prijava bila uspješna preusmjeravamo se na glavni ekran aplikacije (slika 3.4). Na glavnome (*Home*) ekranu možemo vidjeti popis svih scena, a također možemo filtrirati scene pomoću tagova. Osim toga, nudi nam se mogućnost pretraživanja scena po nazivu. Popis scena dobivamo GET zahtjevom na REST poslužitelj. Klikom na neku scenu otvara nam se ekran na kojemu možemo vidjeti detalje te scene, tj. Prikazuje nam se slika koja predstavlja tu scenu te *viewovi* scene (slika 3.5.). Klikom na neki *view* iz scene otvara nam se graf za odabrani view (slika 3.6.). Podatke za graf dobivamo GET zahtjevom na REST poslužitelj, a iz odgovora se formira novi zahtjev na bazu. Dohvaćeni podaci se potom parsiraju iz csv formata te se prikazuju u obliku grafa. Početni vremenski raspon u kojem se prikazuje graf, a koji dobivamo klikom na *view* je 30 dana. Taj vremenski raspon se može mijenjati pomoću padajućeg izbornika na vrhu ekrana: nudi nam se još raspon od 7 i od jednog dana. Ukoliko odaberemo neku novu opciju šalje se ponovno novi zahtjev na bazu te se potom mijenja i graf (slika 3.8.). Osim grafa na donjoj navigacijskoj traci (engl. bottom down bar) možemo izabrati i opciju prikaza tablice za izabrani *view.* Tablica prikazuje datume za vremenski period u kojem je prikazan graf te vrijednost za svaki datum (x i y koordinate grafa) (slika 3.7.). U slučaju da promjene vremenskog perioda na grafu mijenja se i tablica (slika 3.9.).



*Slika 3.2.1. početni zaslon Slika 3.2.2. Server Settings Slika 3.2.3. Sign In forma*



*Slika 3.2.4. Home Screen Slika 3.2.5. detalji scene Slika 3.2.6. Prikaz grafa*

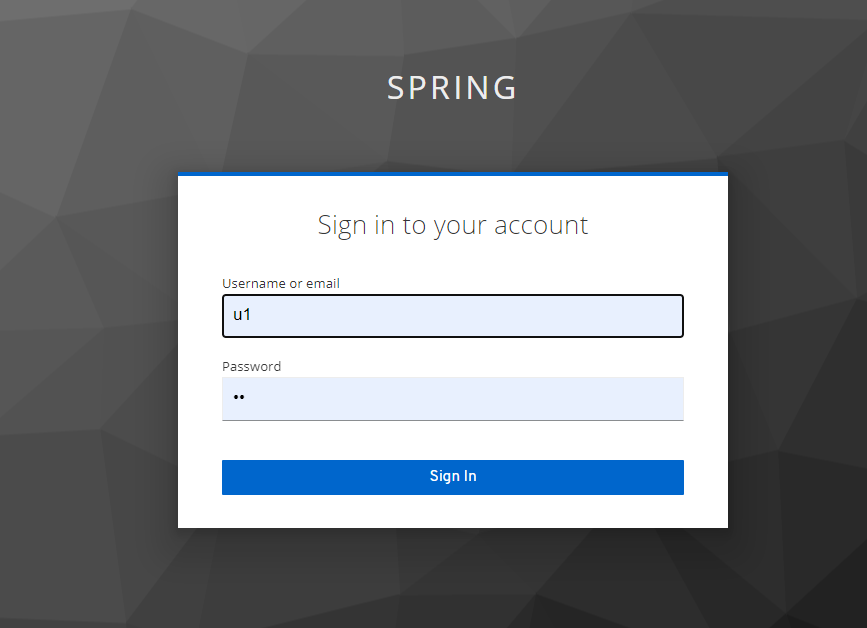
 

*Slika 3.2.7. Prikaz tablice Slika 3.2.8. Graf nakon promjene Slika 3.2.9. Tablica nakon promjene*

*vremenskog perioda vremenskog perioda*

## Web aplikacija

1. **Korak** Prije svega obavezno pokrenuti poslužitelja odnosno backend server !
2. **korak -** prijava na Keycloak serveru. Prilikom uspješne prijave povratno se dobije token i refresh token. Implementirano je da se pri svakom pozivu na backend u headeru šalje token kako bi se održala razina sigurnosti. Bez toga korisnik ne bi mogao pristupiti podatcima s backend servera.



Slika 3.3.1. Prikaz prijave u sustav

1. **korak -** prilikom uspješne prijave korisniku se prikazuje “Home page” stranica. Opcije koje se korisniku ponuđene se nalaze u navigacijskoj traci na vrhu stranice. Opcije su: povratak na početnu stranicu odnosno “POČETNA STRANICA”, prikaz scena odnosno “SCENE” i prikaz ključeva “KLJUČEVI”

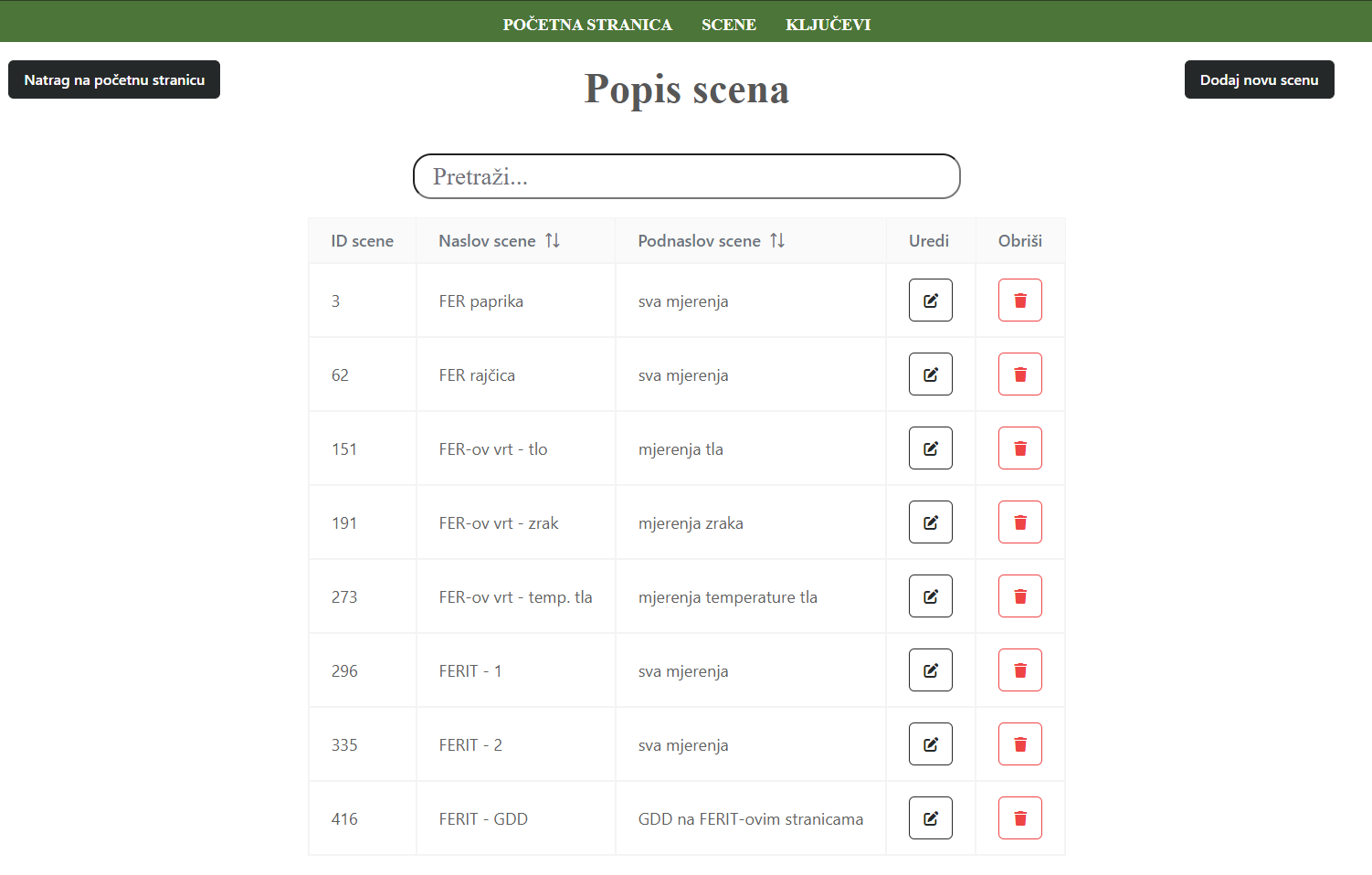


Slika 3.3.2. Početna stranica

1. **korak -** odabir opcije za prikaz scena klik na gumb “SCENE”

Opcije na ovom prikazu:

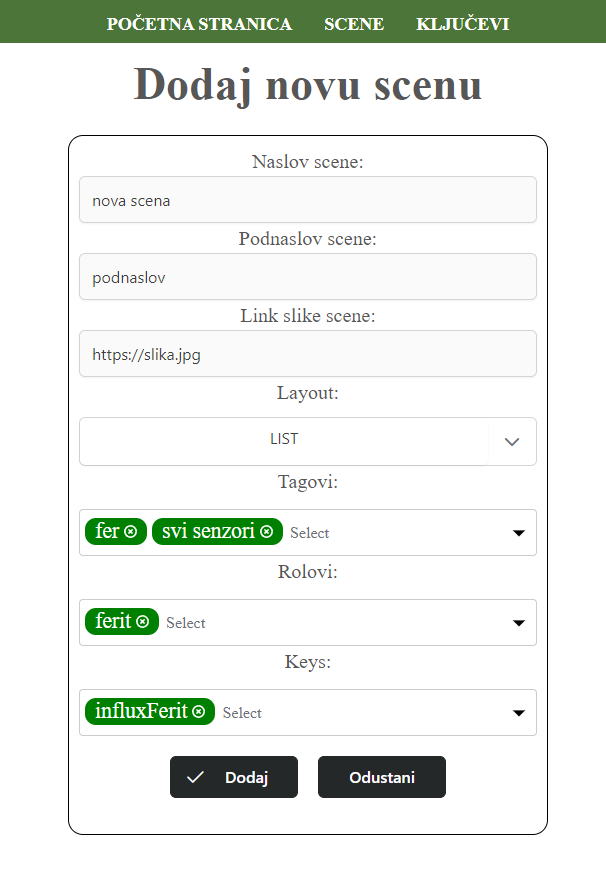
* Povratak na početnu stranicu
* Pretraživanje scena (filtriranje kako bi lakše pronašli traženu scenu)
* Dodavanje nove scene
* Uređivanje scene
* Brisanje scene



Slika 3.3.3. Popis scena

**3. a -** dodavanje nove scene i uređivanje postojeće scene

Primjer forne prilikom dodavanje nove scene te kasnije uređivanje iste ali tek nakon kreiranja.



Slika 3.3.4 Dodavanje scene Slika 3.3.5. Uređivanje scene

**3. b -** brisanje odabrane scene

Prikazani prozorčić (eng. popup) je dodan iz sigurnosnih razloga zbog brisanja velikog broja podataka.



Slika 3.3.6. *Popup* za potvrdu brisanja

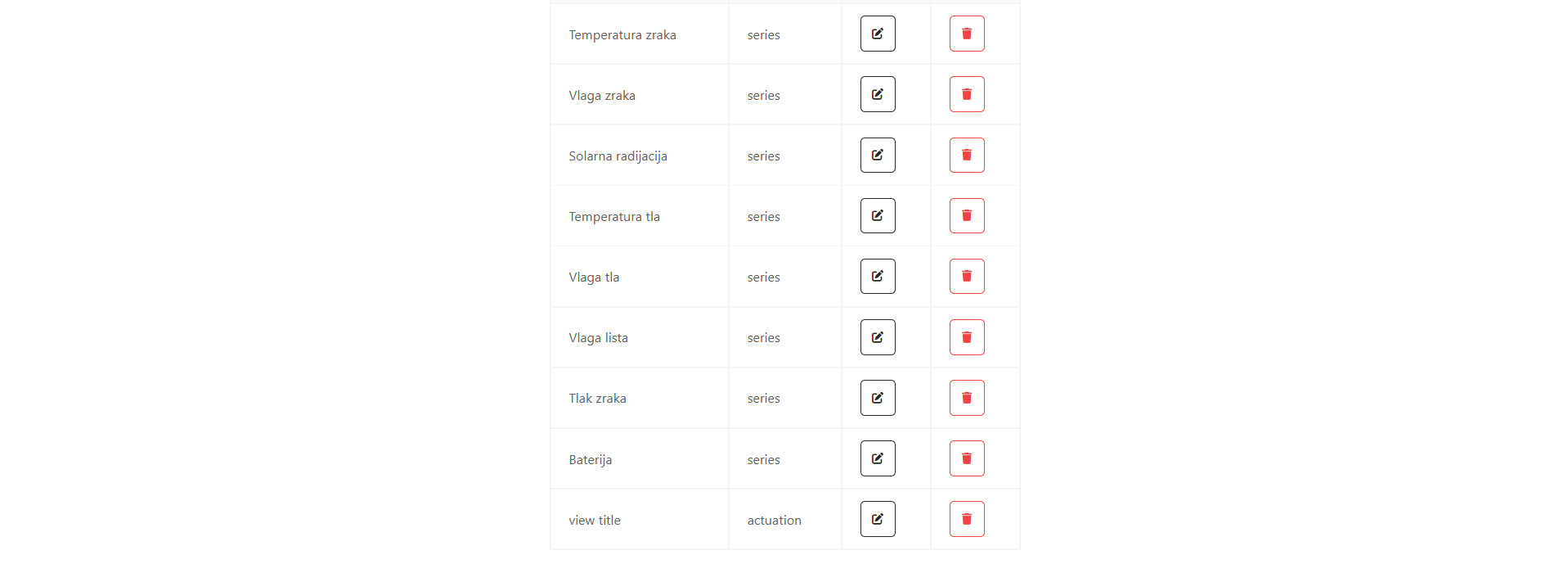
**3. c -** odabir specifične scene unutar tablice (klik na redak scene za koju želimo vidjeti više detalja)

Na ovom prikazu je također omogućeno uređivanje scene ili njeno brisanje. Ispod slike odabrane scene (u ovom slučaju FER paprika) se nalazi popis view-ova za pripadnu scenu koji se nalaze unutar tablice kao i kod prikaza scena.

Graphical user interface, text

Description automatically generated

Slika 3.3.7. Pregled scene FER paprika



Slika 3.3.8. Popis view-ova za odabranu scenu

1. **korak –** upravljanje podatcima view-a za odabranu scenu

Opcije za listu view-a:

* Dodavanje novog view-a (2 opcije): “Add actuation view” ili “Add measurement view”
* Uređivanjem odabranog view-a klikom na gumb “Uredi” unutar tablice
* Brisanje view-a klikom na gumb “Obriši” unutar tablice

**4.a -** dodavanje novog *view*-a

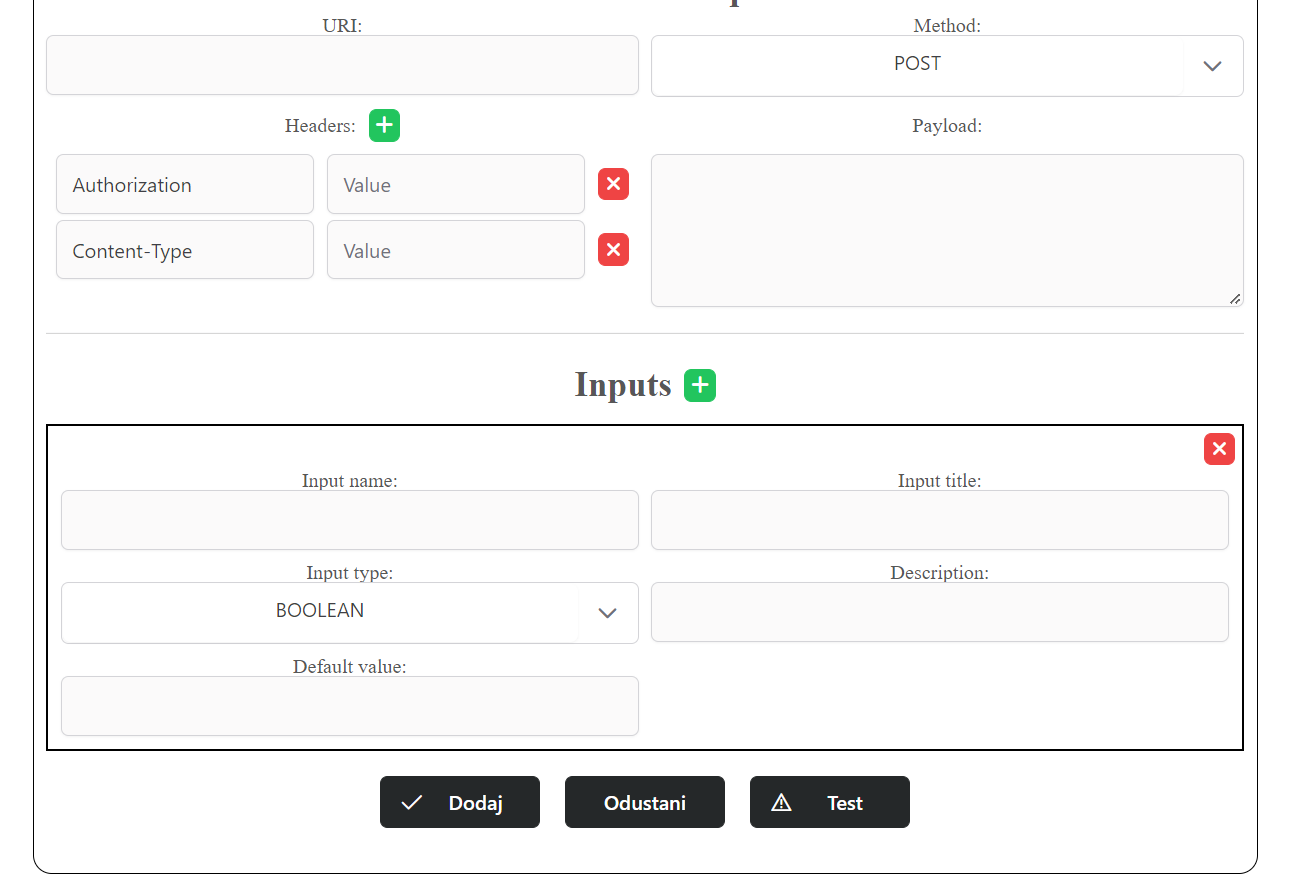
Prva opcija: dodavanje *actuation view-a*.

Prilikom unosa podataka preporučeno je prvo testirati ispravnost unesenih podataka klikom na gumb “Test” te ovisno o odgovoru dodati novi *view* klikom na gumb “Dodaj” ili odustati klikom na gumb “Odustani”.

Graphical user interface, application

Description automatically generated

Slika 3.3.9. Dodavanje *actuation view-a*



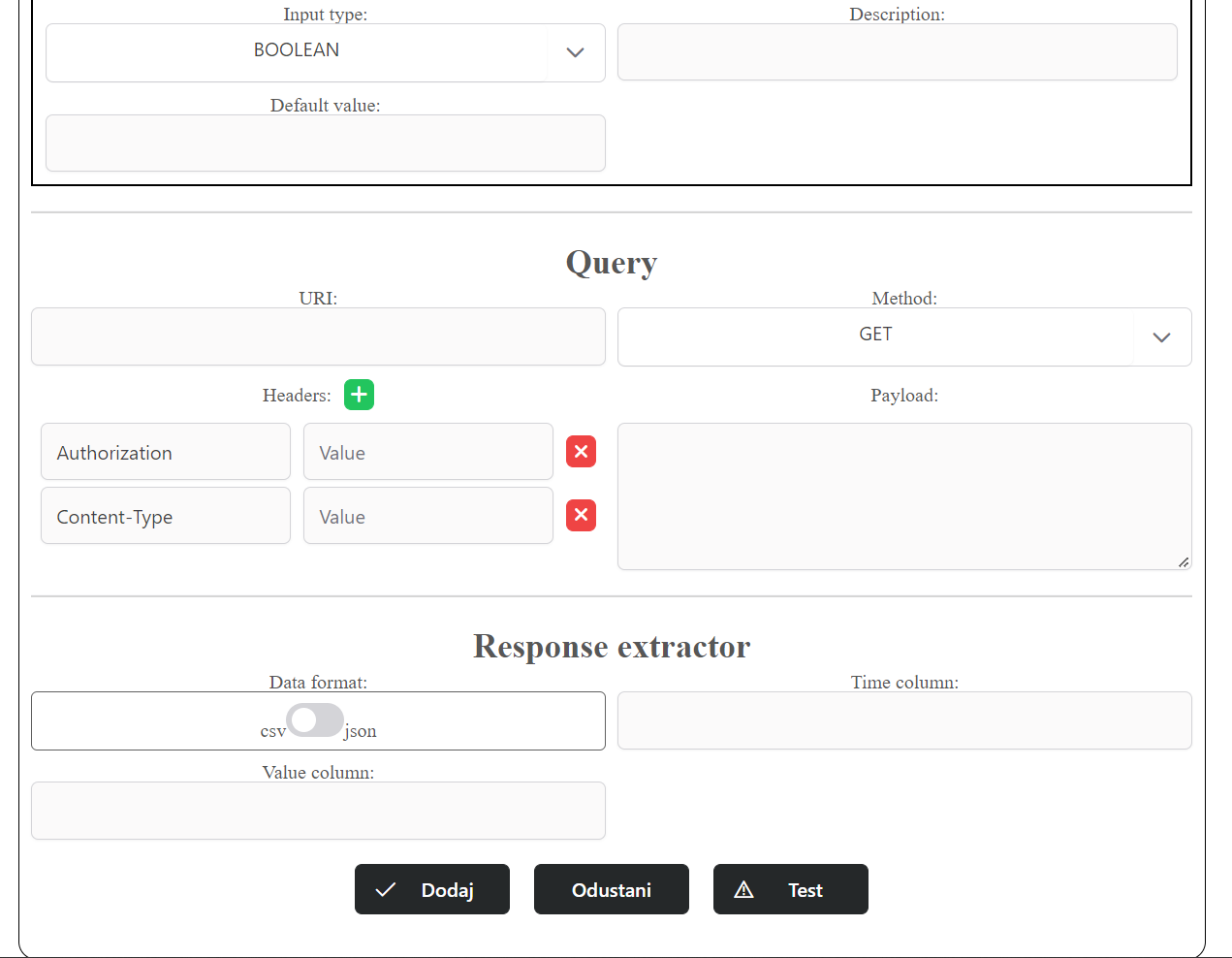
Slika 3.3.10. Dodavanje *actuation viewa* (donji dio ekrana)

Druga opcija je dodavanje measurement view-a. Forma za dodavanje istog izgleda ovako:

Graphical user interface, text, application, email

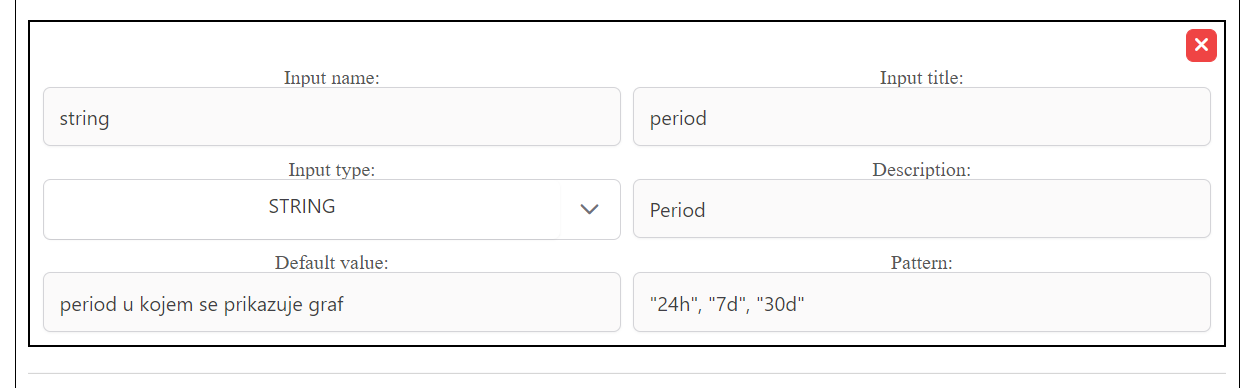
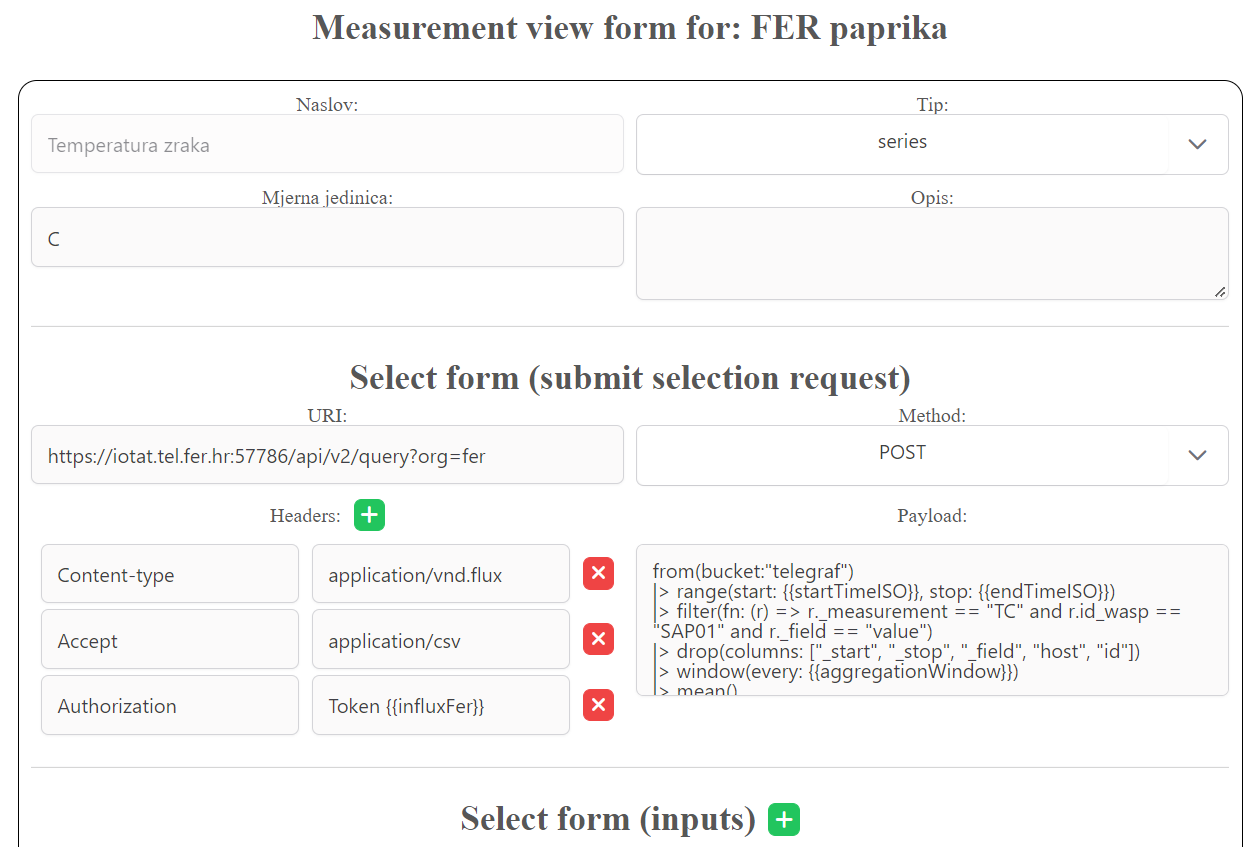
Description automatically generated

Slika 3.3.11. Dodavanje measurement view-a



Slika 3.3.12. Dodavanje measurement view-a (donji dio)

**4. b -** uređivanje odabranog view-a unutar tablice



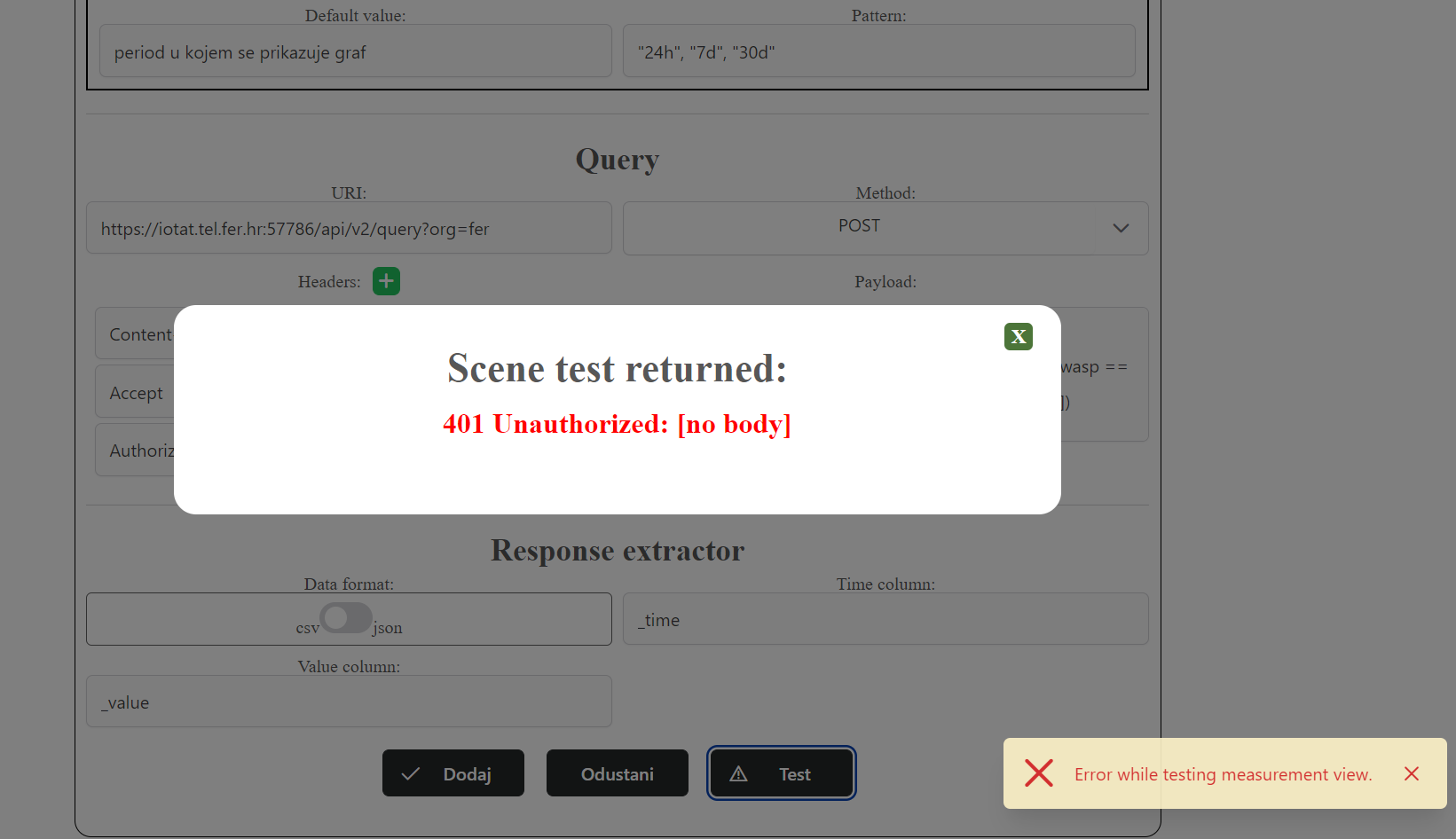
Slika 3.3.13. Uređivanje odabranog view-a



Slika 3.3.14. Uređivanje odabranog view-a (donji dio)

**4. c** - odabir opcije za testiranje

Ukoliko se pojavila pogreška prilikom kreiranja ili uređivanja view-a. Korisnik dobiva jasnu poruku. Primjer poruke koja dojavljuje korisniku gdje je došlo do pogreške.



Slika 3.3.15. Odgovor testiranja scene

**4. d -** odabir opcije za brisanje view-a



Slika 3.3.16. Potvrda brisanja view-a

1. **korak -** odabir opcije za prikaz ključeva klik na gumb “KLJUČEVI”

Opcije na ovom prikazu:

* Povratak na početnu stranicu
* Pretraživanje ključeva (filtriranje kako bi lakše pronašli traženi ključ)
* Dodavanje novog ključa
* Uređivanje ključa
* Brisanje ključa

**5. a** - dodavanje novog ključa i uređivanje postojećeg ključa

Primjer dodavanje novog ključa te kasnije uređivanje istog ali tek nakon kreiranja



Slika 3.3.17. Dodavanje ključa Slika 3.3.18. Uređivanje ključa



Slika 3.3.19. Prikaz novog ključa u tablici

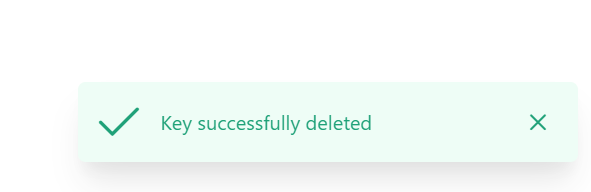
**5. b** - brisanje odabranog ključa



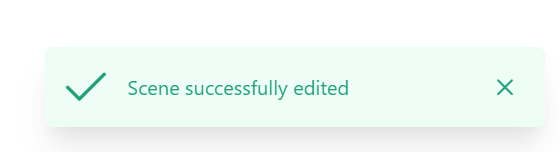
Slika 3.3.20. Potvrda za brisanje ključa

- unosom “OBRIŠI KLJUČ” dobije se poruka sa servera odnosno “toast message”. Ove poruke se pojavljuju svugdje prilikom CRUD (eng create, read, update i delete) akcija za scene, ključeve i view-ove. Time se postiže bolja interakcija korisnika i servera u prilikom korištenja web aplikacije.

- još neki primjeri uspješnih akcija te prikaz poruka sa servera

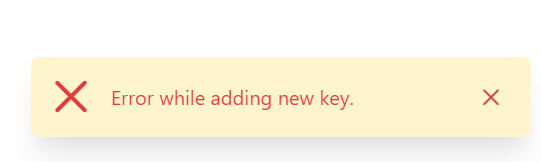


Slika 3.3.21. Potvrda o brisanju ključa



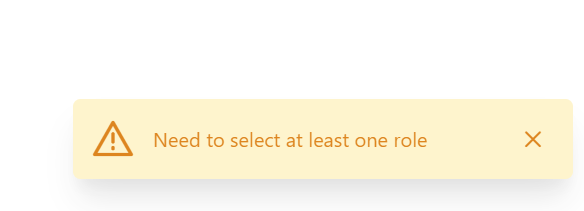
Slika 3.3.22. Potvrda o uređivanju scene

- primjer neuspjele akcije te prikaz poruke



Slika 3.3.23. Poruka o grešci pri dodavanju novog ključa

- prikaz poruke upozorenja



Slika 3.3.24. Poruka upozorenja

## Mikroupravljač, senzori, LoRaWAN Gateway i ChirpStack poslužitelj

### Mikroupravljač i senzori

**Clicker 2 for STM32** upravlja svim senzorima i ostalim dijelovima sustava za navodnjavanje te pomoću LoRaWAN modula komunicira s mrežnim prilazom (eng. *gateway*) i mrežnim poslužiteljem.

**Za pripremu pločice potrebno je napraviti sljedeće:**

1. Na mikroupravljač je potrebno zalemiti (slika 3.4.1.3.) pinove sa slike 3.4.1.2. na koje će se spajati senzori i ostali dijelovi sustava.
2. Za komunikaciju s mrežnim poslužiteljem potrebno je imati spojen/prikačen LR 4 Click, LoRa modul prikazan na slici 3.4.1.4. i finalni rezultat spajanja na slici 3.4.1.5.
3. Ostali senzori spajaju se na pločicu prema slici 3.4.1.6. i 3.4.1.7.
4. U kodu (programski jezik C) potrebno je podesiti varijablu MIKROBUS\_1 ili 2 ovisno o mikroBUS-u na koji smo spojili LR 4 Click (slika 3.4.1.8. prikazuje isječak koda koji je potrebno podesiti)

A picture containing text, electronics, circuit

Description automatically generated

Slika 3.4.1. Shema Clicker2

A picture containing text, electronics

Description automatically generated

Slika 3.4.1.2. Pinovi za Clicker2 for STM32

A picture containing text, electronics, circuit

Description automatically generated

Slika 3.2.1.3. Lemljenje pinova

A picture containing text, electronics

Description automatically generated

Slika 3.4.1.4. LR 4 Click

A green circuit board

Description automatically generated with low confidence

Slika 3.4.1.5. LR 4 Click spojen na Clicker 2 pločici

A circuit board with wires

Description automatically generated with low confidence

Slika 3.4.1.6. Spojeni senzori na pločicu



Slika 3.4.1.8. Podešavanje u kodu

Diagram, schematic

Description automatically generated

Slika 3.4.1.7. Shema sustava (preuzeto iz Gudelj, Petar Šimun, 2022. Prikupljanje podataka i aktuacija s uređajima u urbanom vrtu, Diplomski rad. Fakultet Elektrotehnike i računarstva)

### LoRaWAN Gateway i ChirpStack poslužitelj

Za spajanje na mrežu LoRaWAN potrebno je spojiti se na ChirpStack, LoRaWAN mrežni poslužitelj preko kojega možemo dodavati, brisati i uređivati uređaje (poput Clicker2), gatewaye i još mnogo toga što nije važno trenutno za pokretanje aplikacije.

Na ChirpStacku je već izrađena aplikacija i kofiguriran gateway, a dodavanje uređaja će biti objašnjeno u nastavku.

Za spajanje na ChripStack potrebno je učitati projekt u PuTTY (na slici 3.4.2.1. „projektR22“) i pokrenuti, otvorit će se terminal (slika 3.4.2.2.) koji će tražiti podatke za vašu prijavu kako biste mogli otvoriti ChirpStack na http://localhost:8080.

Kako bi se uređaj dodao potrebno je ući u *Applications*, željenu aplikaciju i kliknuti gumb + CREATE. Tada će biti potrebno ispuniti formu za kreiranje uređaja kao na slici 3.4.2.3. (u nastavku dolje). Da bi se uređaj mogao povezati s poslužiteljem potrebna je OTAA aktivacija koja će se dogoditi nakon unošenja ispravnog aplikacijskog ključa i DevEUI broja u programskom kodu, odnosno kod dodavanja uređaja. (vidi sliku 3.4.2.4.) Osim aplikacijskog ključa trebat će promijeniti način spajanja, ako već nije promijenjen, kako je prikazano na slici 3.4.2.5.

Nakon uspješnog spajanja na kreće se sa uzlaznom komunikacijom, primjer uzlazne poruke možemo vidjeti na slici 3.4.2.6.

Graphical user interface, application

Description automatically generated

Slika 3.4.2.1. PuTTY prozor

Text

Description automatically generated

Slika 3.4.2.2. PuTTY terminal – *login*

Graphical user interface, text, application, email

Description automatically generatedSlika 3.4.2.3. Dodavanje uređaja

Text

Description automatically generated

Slika 3.4.2.3. Dodavanje aplikacijskog ključa u kod

A screenshot of a computer

Description automatically generated with medium confidence

Slika 3.4.2.4. Promjena u kodu

Graphical user interface, text, application

Description automatically generated

Slika 3.4.2.5. Primjer uzlazne poruke

# Literatura

1. Đurinec, Jan i dr. 2022. Mobilno IoT-polje. Tehnička dokumentacija. Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu. Zagreb. 14 str.
2. Sruk, Vlado; Jović, Alan; Frid, Nikolina. 2022. Radni okviri za web razvoj. Materijali s predavanja Programskog inženjerstva. Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu. Zagreb. 82 sl.
3. Keycloak. Dokumentacija. <https://www.keycloak.org/documentation> (pristupljeno 6. siječnja 2023.)
4. User manual [clicker2-stm32-manual-v100.pdf (mikroe.com)](https://download.mikroe.com/documents/starter-boards/clicker-2/stm32f4/clicker2-stm32-manual-v100.pdf) (zadnje pristupljeno 16.01.2023)
5. Gudelj, Petar Šimun, 2022. Prikupljanje podataka I aktuacija s uređajima u urbanom vrtu, Diplomski rad. Fakultet Elektrotehnike i računarstva