Manuali i perdorimit KosovAR

Hyrje

Ky user manual është krijuar është përgatitur me qëllim që të shërbejë si një udhëzues i plotë dhe i qartë për kedo që eshte i përfshirë në mirëmbajtjen, zhvillimin dhe funksionimin e përditshëm të web aplikacionit **KosovAR**. Ai përmban informacion të strukturuar dhe praktik mbi të gjitha aspektet teknike të sistemit, që nga konfigurimi fillestar deri te monitorimi dhe menaxhimi i avancuar i funksioneve të brendshme.Ai përfshin udhëzime të hollësishme për ambientimin e mjedisit të zhvillimit, instalimin, përshkrimin e arkitekturës, menaxhimin e të dhënave, autentifikimit dhe integrimit të augmented reality (AR).

Instalimi dhe konfigurimi

Hapat në vijim paraqesin pjesën e instalimit të repositorit lokalisht prej GitHub:

```
git clone https://github.com/lerdisalihi/Kosovari
cd Kosovari
npm install
npm run dev
```

Teknologjitë e përdorura

Në këtë pjesë do te prezantojme listen e plotë te teknologjive të përdorura në ndërtimin dhe funksionimin e aplikacionit KosovAR. Çdo teknologji është zgjedhur për të adresuar një nevojë të veçantë të sistemit dhe për të siguruar që aplikacioni të jetë modern, i shkallëzueshëm, i sigurt dhe i lehtë për mirëmbajtje.

Frontend-i

KosovAR-i në pjesën e frontendit është i ndërtuar me komponentë të rigjenerueshëm që komunikojnë me backend-in përmes API-ve të Supabase. Ku më poshtë do të tregojmë në detaje të gjitha teknologjitë e përdorura:

React.js + TypeScript

React është libraria kryesore për ndërtimin e ndërfaqes së përdoruesit në KosovAR. Gati çdo pamje dhe ndërveprim – nga faqja kryesore, te raportimi i një problemi, e deri te modalet për komentim – janë ndërtuar me React.

- Përdoret për:
    - Strukturimin e faqes në komponente (p.sh. IssueCard.tsx, IssueReportDialog.tsx, ARQRCodeModal.tsx)
    - Ndërtimin e formave për login/signup
    - Komunikimin në faqe përmes React Router
- TypeScript shton tipizim të sigurt dhe na ndihmon të parashikojmë dhe shmangim gabime logjike.

Shembull në projekt:

```
const IssueCard = ({ issue }: { issue: Issue }) => ( <div>{issue.description}</div> );
```

Tailwind CSS

Tailwind është përdorur për të dizajnuar aplikacionin pa pasur nevojë për file-ës të ndarë CSS. Me klasat e gatshme, kemi te dizajnuar UI moderne, responsive dhe të qartë për përdoruesin.

- Përdoret për:
    - Stilizimin e elementeve UI në mënyrë të menjëhershme (bg-white, rounded-lg, text-gray-800)
    - Responsivitet në pajisje mobile
    - Strukturim të kartave të problemeve, modaleve, formularëve dhe dashboard-it admin

Shembull në projekt:

```
<div className="bg-white p-4 rounded-lg shadow-md">

  <h2 className="text-xl font-bold">Raporto problem</h2>

</div>
```

Vite

Vite është përdorur si build tool për të përpiluar kodin frontend me shpejtësi shumë të lartë, krahasuar me Webpack.

- Përdoret për:
    - Shërbim të aplikacionit në zhvillim lokal (vite dev server)
    - Build final për deploy në Vercel (vite build)
    - Menaxhimin e environment variables (import.meta.env)

Shembull përdorimi në projekt:

```
const supabaseUrl = import.meta.env.VITE_SUPABASE_URL;
```

Backend

Backend-i në KosovAR nuk është një server tradicional, por përdor Supabase si një platformë “Backend-as-a-Service” që ofron auth, ruajtje të të dhënave dhe storage në cloud.

Supabase

Supabase(Link) është zgjedhur si backend i tërë aplikacionit KosovAR. Ai kombinon autentifikimin, bazën e të dhënave PostgreSQL, storage për media dhe RLS (Row-Level Security).

- Përdoret për:
    - Autentifikim të përdoruesve (signUp, signIn, session)
    - Ruajtje të raporteve të problemeve në issues (kategori, përshkrim, koordinata)
    - Ngarkim të imazheve të problemeve në storage
    - Politika sigurie që lejojnë vetëm autorin të shohë/edit raportin e tij (auth.uid() = user_id)
    - Kontrollin e roleve (citizen vs. institution)

Shembull në projekt:

```
const { data, error } = await supabase

 .from('issues')

 .insert([{ description: "Dritë e thyer", latitude, longitude }]);
```

Zustand

Zustand është përdorur për të menaxhuar gjendjen globale të aplikacionit pa komplikime si Redux. Ai është veçanërisht i përshtatshëm për KosovAR për shkak të kompleksitetit relativisht të ulët të logjikës së brendshme.

- Përdoret për:
    - Menaxhimin e listës së issues (problemeve)
    - Ndryshimin e gjendjes së raportimit (isReportingMode)
    - Operacione si addIssue, deleteIssue, updateIssueStatus

Shembull në projekt:

```
addIssue: async (newIssue) => {

 const { data } = await supabase.from('issues').insert(newIssue);

 set(state => ({ issues: [...state.issues, data[0] ] }));

}
```

Hostimi dhe menaxhimi i sistemit

Netlify

Netlify është platformë moderne për hostimin e aplikacioneve frontend statike, si ato të ndërtuara me React + Vite etj. Ajo ofron një përvojë shumë të thjeshtuar për deploy dhe integrim me GitHub, si dhe një sërë funksionalitetesh që automatizojnë ndërtimin dhe publikimin e përmbajtjes në web.

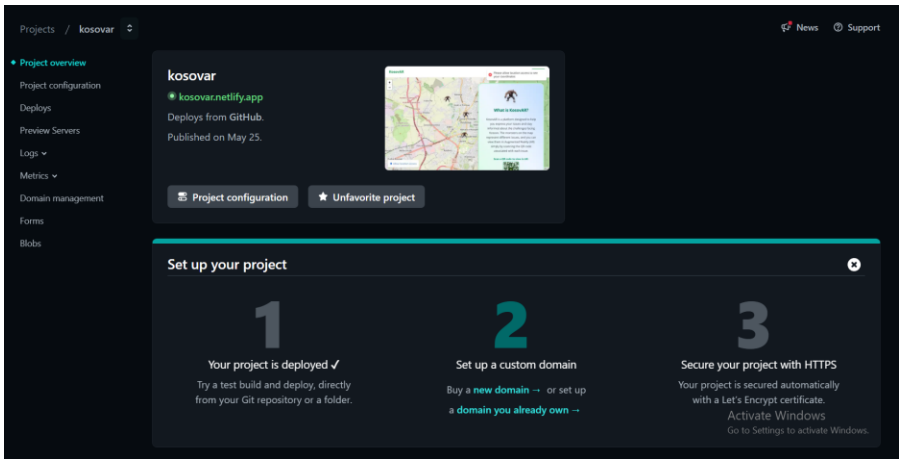
Përdorimi i Netlify në KosovAR:

Në projektin KosovAR, Netlify përdoret për të:

- Hostuar **build-in e frontend-it** (React/Vite)
- Siguruar **HTTPS automatik me SSL**
- Menaxhuar **CI/CD përmes GitHub** (çdo push në main → deploy)
- Ofruar **preview deployments** për çdo pull request që krijohet

Rrjedha e punës me Netlify:

1. Mirëmbajtësi bën push në GitHub.
2. Netlify automatikisht detekton ndryshimin dhe ekzekuton:
    a. npm install
    b. npm run build
3. Dosja dist/ e krijuar nga Vite publikohet në CDN-in e Netlify.
4. Aplikacioni bëhet i aksesueshëm përmes një domain-it: https://kosovar.netlify.app



Supabase Dashboard

Supabase Dashboard është paneli ku menaxhohet backend-in e aplikacionit KosovAR. Ai shërben si zëvendësues i një backend-i tradicional dhe përfshin:

1. Tables & Data Explorer-Përdoret për të parë dhe menaxhuar të dhënat në tabelat users, issues, comments, etj. Lejon inspektim, filtrim dhe shtim manual të të dhënave.

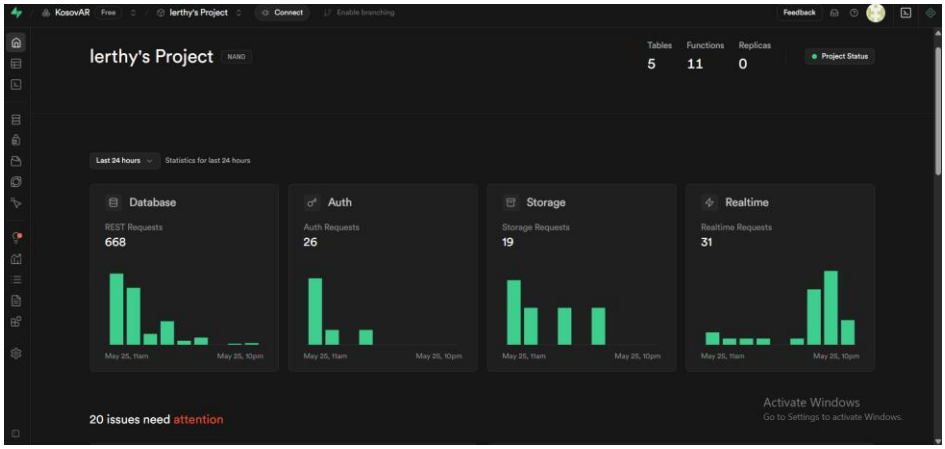
2. Authentication-Shfaq listën e përdoruesve të regjistruar. Mund të shihen email-et, rolet dhe metadata, si dhe të ndryshohen manualisht.

3. Storage-Përdoret për të ruajtur imazhet e problemeve të ngarkuara nga përdoruesit. Adminët mund të shohin dhe fshijnë skedarë nga folderi public.

4. Policies (RLS)-Kontrollojnë qasjen e përdoruesve te të dhënat. Në KosovAR, vetëm përdoruesi që ka krijuar një raport ose një institucion mund ta editojë ose shohë atë.

5. Logs & SQL-Lejon debug të operacioneve (login, insert, update) dhe krijimin e queries për testim apo statistika.

Arsyeja pse është efektive përdorimi i saj është sepse mundëson menaxhim të plotë të të dhënave, qasjes, mediave dhe sigurisë – pa nevojën e ndërtimit të një backend-i të dedikuar nga e para.



ARKITEKTURA E SISTEMIT

Arkitektura e aplikacionit **KosovAR** është e ndarë në tri shtresa kryesore, të ndërtuara sipas parimeve të ndarjes së përgjegjësive (**Separation of Concerns**) dhe modularitetit. Kjo qasje ndihmon në mirëmbajtjen, testimin dhe shkallëzimin e aplikacionit.

Shtresa e Parë: Frontend

Frontend-i është ndërtuar me **React.js + TypeScript**, ku çdo funksionalitet përfaqësohet me komponentë të ndarë dhe të ripërdorshëm.

Përgjegjësitë:

- Menaxhimi i ndërveprimeve të përdoruesit (formularë, klikime, QR scanner)
- Vizualizimi i përmbajtjes dinamike (karta të problemeve, komentet)
- Dërgimi i të dhënave në backend përmes Supabase SDK
- Marrja e të dhënave në kohë reale dhe rifreskimi i UI-së

Komponentët kyç:

- IssueReportDialog.tsx – Formular për krijimin e raportit
- IssueCard.tsx – Komponent për shfaqje të problemeve të raportuara
- ARQRCodeModal.tsx – Aktivizon kamerën dhe lexon QR code për AR
- Header.tsx, Footer.tsx – Komponentë të përbashkët të layout-it

Shtresa e Dytë: Store & Logjikë e Aplikacionit

Kjo shtresë është përgjegjëse për **menaxhimin e gjendjes së aplikacionit** dhe për ndërveprimin me backend-in në mënyrë të qëndrueshme dhe të kontrolluar.

Teknologjia:

- Përdoret **Zustand** për të krijuar store-e funksionale dhe të lehta për përdorim

Funksionet kryesore:

- fetchIssues() – Merr të gjitha problemet nga Supabase
- addIssue(data) – Shton problem të ri në DB dhe rifreskon gjendjen
- updateIssueStatus(id, status) – Ndryshon statusin e një problemi ekzistues
- deleteIssue(id) – Fshin problem nga DB dhe UI

Skedarët:

- src/store/issues.ts – Gjendja dhe funksionet për raportet
- src/store/auth.ts – Gjendja e përdoruesit të kyçur dhe roli i tij

Shtresa e Tretë: Backend

Supabase është zgjedhur si backend cloud për shkak të integrimit të Auth, Database dhe Storage në një platformë të vetme dhe të menaxhueshme pa server të dedikuar.

Komponentët:

- **Supabase Auth** – për autentifikim me email/password dhe menaxhim të sesioneve
- **Supabase PostgreSQL** – ruan të gjitha të dhënat (raporte, përdorues, komente, vota)
- **Supabase Storage** – ruan mediat që përdoruesit ngarkojnë (foto të problemeve)
- **RLS (Row-Level Security)** – siguron që çdo përdorues sheh vetëm të dhënat që i përkasin atij ose sipas rolit të caktuar

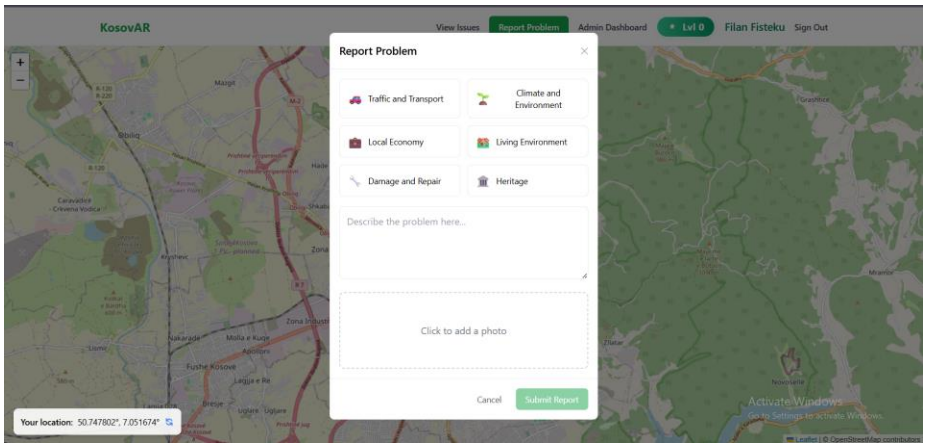
Skedari kryesor për komunikim:

- src/lib/supabase.ts – Krijon një instancë të klientit Supabase për përdorim në të gjithë aplikacionin

Rrjedha e të Dhënave

1. **Përdoruesi hap modalin** për të raportuar një problem (IssueReportDialog)
2. Plotëson përshkrimin, zgjedh kategorinë, vendos lokacionin (përmes browser geolocation) dhe opsionalisht ngarkon një imazh
3. Në klikim të "Raporto", aplikacioni:
    a. Dërgon të dhënat përmes funksionit addIssue() në useIssueStore
    b. Kjo funksion përdor Supabase SDK për të bërë një insert në tabelën issues
    c. Nëse ka media, ajo ngarkohet në supabase.storage.from('public').upload(...)
    d. URL e medias vendoset në fushën image_url të tabelës
4. Nëse gjithçka shkon mirë:
    a. Gjendja rifreskohet me fetchIssues()
    b. UI shfaq problemin e ri menjëherë për përdoruesin dhe për të tjerët (varësisht nga roli)
5. Të gjitha këto veprime janë të mbrojtura përmes:
    a. JWT token të përdoruesit të kyçur

b. RLS që kufizon shikimin dhe modifikimin sipas auth.uid() dhe user_metadata.role



Shembull në projekt:

```
const { data, error } = await supabase.from('issues').insert({
  category: values.category,
  description: values.description,
  image_url: imageUrl,
  latitude,
  longitude,
  user_id: user.id,
});
```

Përfitimet nga kjo arkitekturë janë:

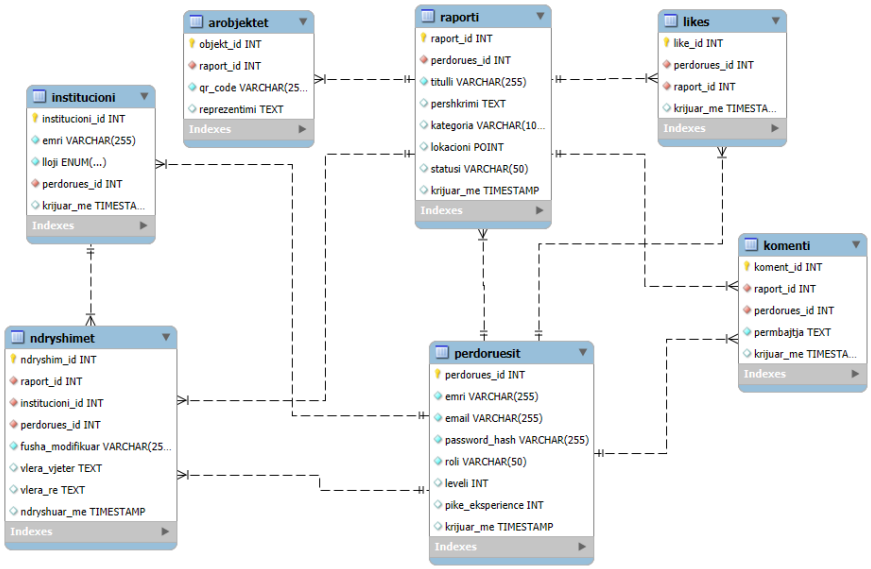
- **Shkallëzueshmëri**: Frontend dhe backend janë të ndarë dhe mund të rriten në mënyrë të pavarur
- **Modularitet**: Çdo komponent ka një rol të qartë (UI, gjendje, backend)
- **Siguri**: Çdo qasje është e kontrolluar me RLS dhe autentifikim
- **Shpejtësi zhvillimi**: Supabase e zvogëlon nevojën për backend të ndërtuar me dorë

Baza e të dhënave

Aplikacioni KosovAR përdor një bazë të të dhënave të ndërtuar mbi **PostgreSQL**, e menaxhuar përmes **Supabase**, me fokus në modularitet, siguri dhe qartësi të strukturës. Supabase ofron gjithashtu funksione të integruara për autentifikim dhe ruajtje të skedarëve, por pjesa kryesore e logjikës së të dhënave ruhet në

tabelat e mëposhtme.
ku shohim

Mbështetje dhe kontakt teknik



Në rast se keni nevojë për ndihmë teknike apo rikuperim të të dhënave knontaktoni ne emalin tonë support@kosovarapp.com ose në numrin tonë të telefonit: +383 44 505 000

Ekipa jonë është në dispozicion 24/7 për mirëmbajtje dhe asistencë emergjente.