Arhitekturne odločitve  
  
Helper services:

Narejene so v pythonu, saj ponuja ogromno knjižnic za obdelavo datotek in delo s podatki, kar pa je pri nas bilo ključnega pomena. Omogoča tudi hitro in enostavno razvijanje rešitev. Prav tako pa je tudi odlična izbira za machine learning, saj že ima knjižnice, ki omogočijo hitrejše učenje modela(ni potrebe po razvoju od začetka).

Backend

Logika aplikacje je napisana v Javi. Izbrali smo SpringBoot ogrodje. SpringBoot smo izbrali, saj omogoča enostavno vzpostavitev projekta, potrebuješ le spring-boot-starter-web dependency in že imaš REST API. Springboot ima že vgrajen strežnik, kar je tudi ena izmed prednosti, saj ne potrebujemo ročno nemstiti strežnika. Prav tako pa omogoča odlično podporo s testi, saj lahko vključimo enostavne Junit teste. Sprotno testiranje je ključno za CI/CD, saj moramo zagotoviti da aplikacija deluje preden se deploya. Prav tako omogoča enostavno integracijo drugih sistemov, recimo baza Firestore. Ena od ključnih dejavnikov pa je bila poznavanje jezika – vsi v skupini poznamo in razumemo javo, zato smo menili, da je najboljša izbira.

REST(Representational State Transfer)

Za ta arhitekturni slog smo se odločili, saj smo ga bolj vajeni kot kakšno SOA. REST omogoča gradnjo spletnih API-jev. URLJI so enostavni, uporabljajo HTTP metode (GET, POST, PUT, DELETE). Prav tako pa je brez stanja(**Stateless**), kar omogoča hitrejše odgovore in lažjo komunkacijo, saj si strežnik re potrebuje zapomniti s kom komunicira, ampak to prejvme v vsaki zahtevi. Prav tako pa je pomembno omeniti **HATEOS**, ki je del REST. HATEOS omogoča uporabniku, da ne potrebuje vedeti naslednjih akcij na pamet, ampak ko dobi odgovor od strežnika le klika po naslednjih povezavah(vidi, kaj so lahko naslednej povezave, ki jih lahko obišče). **Client-Server**, odgovornost je ločena. Backend in frontend sta ločena ampak komunicirata preko API-ja. **Uniform interface**, enoten vmesnik – prednost saj vsi deli sistema komunicirajo preko HTTP metod, URI-ji pa so standarizirani.

Frontend: React

React omogoča razvoj UI s komponentami, kar po našem mnenju poveča preglednost in enostavnost razvoja. React pdopira ogromno knjižnic. Odličen je za razvoj dinamičnih in odzivnih strani. Omogoča uporabo stanj in use effect(kar nam je omogočilo da smo pridobili določene podatke uporabnika in jih shranili v stanje).

Frontend: Vite

Vite omogoča hitro zaganjanje in odstrani potrebo po ponovnem zagonu aplikacije vsakič ko je nekaj spremenjeno, kar prihrani čas. Omogoča enostavno integracijo s React in TS.

Frontend: TypeScript

Je kot JS, vendar nam omogoča določiti tip spremenljivke, kar doda k varnosti(zagotovi, da bo spremenljivka tipa, ki ga določimo). Prav tako pa je poznan nam vsem in smo z njim največ delali v mešanici z React.

Podatkovna baza – Firestore

Firestore je odličen, saj je zelo hiter in ponuja do neke mere zastonj plan. Omogočil nam je hiter razvoj, enostavno ga je integrirati z backend in frontend (springboot + react). Omogočil nam je shranjevanje uporabnikovih računov v google auth bazo, kar pa je bilo ključnega pomena pri varnosti, saj nam je omogočal enostavno integracijo z google 2FA - TOTP. Občutljive podatke pa prav tako sam po sebi hashira ob registraciji. Firestore nam omogoča kasnejšo skalabilnost(horizontalno skaliranje), v primiru, da bi aplikacija zrastla, ker omogoča plačilo po porabi, torej ni fiksne cene, ampak je dogovorjena glede na število zahtev in odogvorov z bazo(read, write, delte).

NOSQL

Razmišljali smo o PostgreSQL ali pa MySQl, vendar smo prišli do ugotovitve, da nekateri imajo sončno elektrarno, drugi pa ne. To vpliva na strukturo dokumentov(XLSX ali CSV), ki se pridobijo iz MojElektro in se naložijo na našo aplikacijo. Tisti ki imajo sončne celice imajo oddano energijo, tisti ki pa nimajo sončnih celic pa imajo vrednost oddane energije 0. To bi v navadni SQL bazi pomenilo nepotrebne zapise v obliki 0 ali pa NULL. Kar pa nam NOSQL odpravi. V primeru da uporabnik nima sončnih celic, to vrednost potem samo izpustimo v dokumentu, namesto da jo dodatno pišemo.