FIIT STU
2024/25

Dokumentácia k semestrálnemu projektu z predmetu VPWA

Autori: Dominik Tulach, Tomáš Ján Černega

Obsah

Zadanie

Vytvorte progresívnu webovú aplikáciu na textovú komunikáciu v štýle IRC (Slack), ktorá komplexne rieši nižšie definované prípady použitia.

Prípady použitia (use cases)

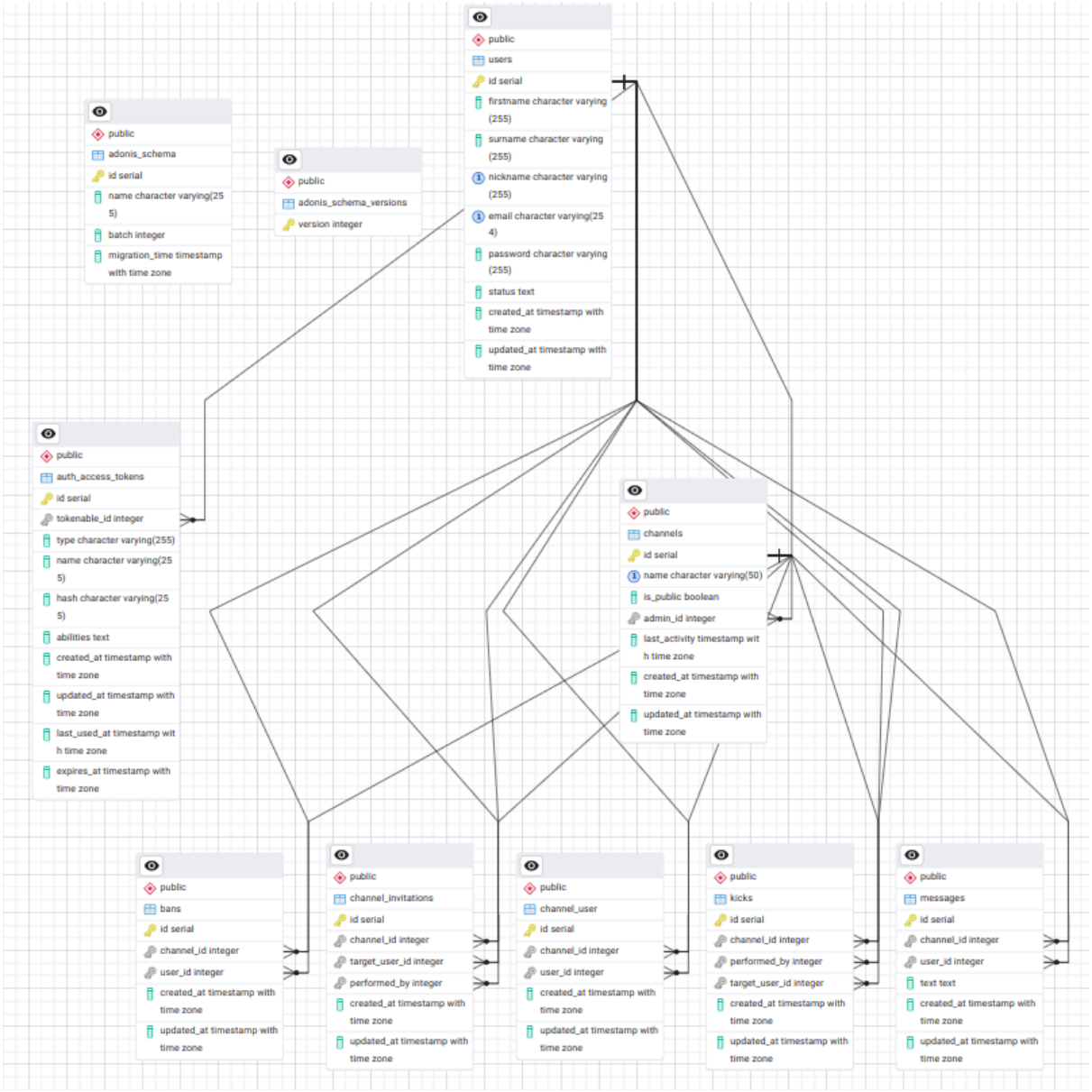
1. registrácia, prihlásenie a odhlásenie používateľa
    a. používateľ má meno a priezvisko, nickName a email
2. používateľ vidí zoznam kanálov, v ktorých je členom
    a. pri opustení kanála, alebo trvalom vyhodení z kanála je daný kanál odobratý zo zoznamu
    b. pri pozvánke do kanála je daný kanál zvýraznený a topovaný
    c. v zozname môže cez používateľské rozhranie kanál vytvoriť, opustiť, a ak je správcom aj zrušiť
    d. dva typy kanálov - súkromný (private channel) a verejný kanál (public channel)
    e. správcom kanála je používateľ, ktorý kanál vytvoril
    f. ak nie je kanál aktívny (nie je pridaná nová správa) viac ako 30 dní, kanál prestáva existovať (následne je možné použiť channelName kanála pre "nový" kanál)
3. používateľ odosiela správy a príkazy cez "príkazový riadok", ktorý je "fixným" prvkom aplikácie. používateľ môže odoslať správu v kanáli, ktorého je členom
4. vytvorenie komunikačného kanála (channel) cez príkazový riadok
    a. kanál môže vytvoriť ľubovolný používateľ cez príkaz /join channelName [private]
    b. do súkromného kanála môže pridávať/odoberať používateľov iba správca kanála cez príkazy /invite nickName a /revoke nickName
    c. do verejného kanála sa môže pridať ľubovolný používateľ cez príkaz /join channelName (ak kanál neexistuje, automaticky sa vytvorí)
    d. do verejného kanála môže člen kanála pozvať iného používateľa príkazom /invite nickName
    e. vo verejnom kanáli môže člen "vyhodiť" iného člena príkazom /kick nickName. ak tak spravia aspoň 3 členovia, používateľ má "trvalý" ban pre daný kanál. správca môže používateľa vyhodiť "natrvalo" kedykoľvek príkazom /kick nickName, alebo naopak "obnovit" používateľovi prístup do kanála cez príkaz /invite
    f. nickName ako aj channelName sú unikátne
    g. správca môže kanál zatvoriť/zrušiť príkazom /quit
5. používateľ môže zrušiť svoje členstvo v kanáli príkazom /cancel, ak tak spraví správca kanála, kanál zaniká
6. správu v kanáli je možné adresovať konkrétnemu používateľovi cez príkaz @nickname
    a. správa je zvýraznená danému používateľovi v zozname správ
7. používateľ si môže pozrieť kompletnú históriu správ
    a. efektívny inifinite scroll
8. používateľ je informovaný o každej novej správe prostredníctvom notifikácie

   a. notifikácia sa vystavuje iba ak aplikácia nie je v stave "visible" (pozrite quasar docu App Visibility)
   b. notifikácia obsahuje časť zo správy a odosielateľa
   c. používateľ si môže nastaviť, aby mu chodili notifikácie iba pre správy, ktoré sú mu adresované
9. používateľ si môže nastaviť stav (online, DND, offline)
   a. stav sa zobrazuje používateľom
   b. ak je nastavený DND stav, neprichádzajú notifikácie
   c. ak je nastavený offline stav, neprichádzajú používateľovi správy, po prepnutí do online sú kanály automaticky aktualizované
10. používateľ si môže pozrieť zoznam členov kanála (ak je tiež členom kanála) príkazom /list
11. ak má používateľ aktívny niektorý z kanálov (nachádza sa v okne správ pre daný kanál) vidí v stavovej lište informáciu o tom, kto aktuálne píše správu (napr. Ed is typing)
   a. po kliknutí na nickName si môže pozrieť rozpísaný text v reálnom čase, predtým, ako ju odosielateľ odošle (každá zmena je viditeľná) :-)

Použité technológie:

- Rámec **Quasar v3** na frontendové šablóny.
- Rámec **AdonisJS v6** na backendové služby.
- DBMS **PostgreSQL** na vytvorenie a prevádzkovanie relačnej databázy.

Dátový model a zmeny oproti 1. fáze

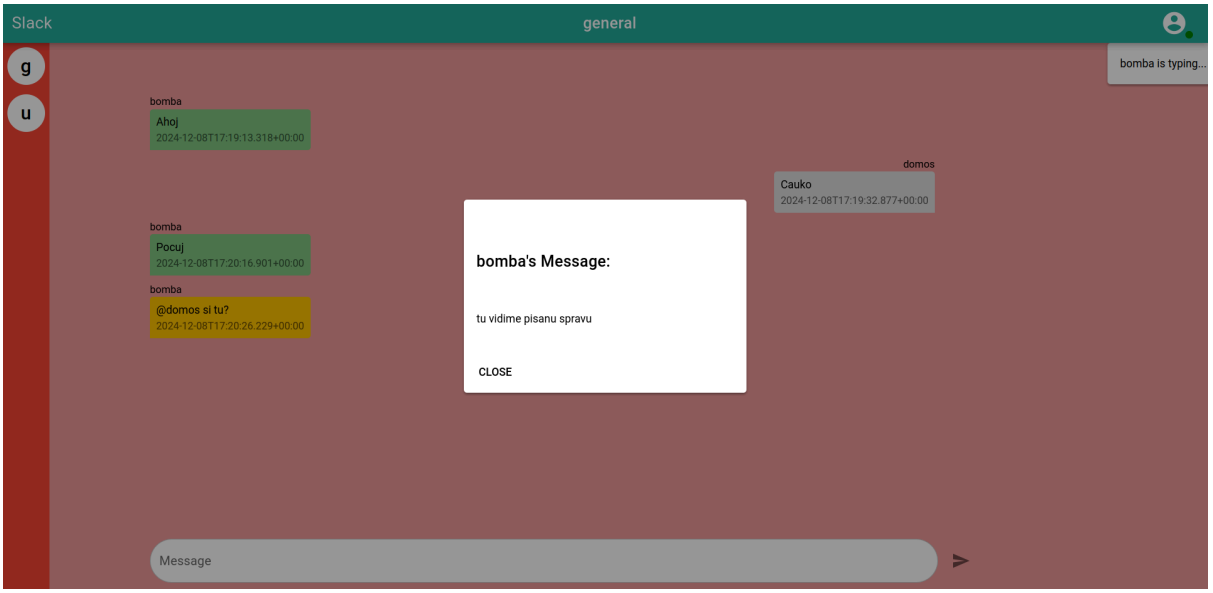


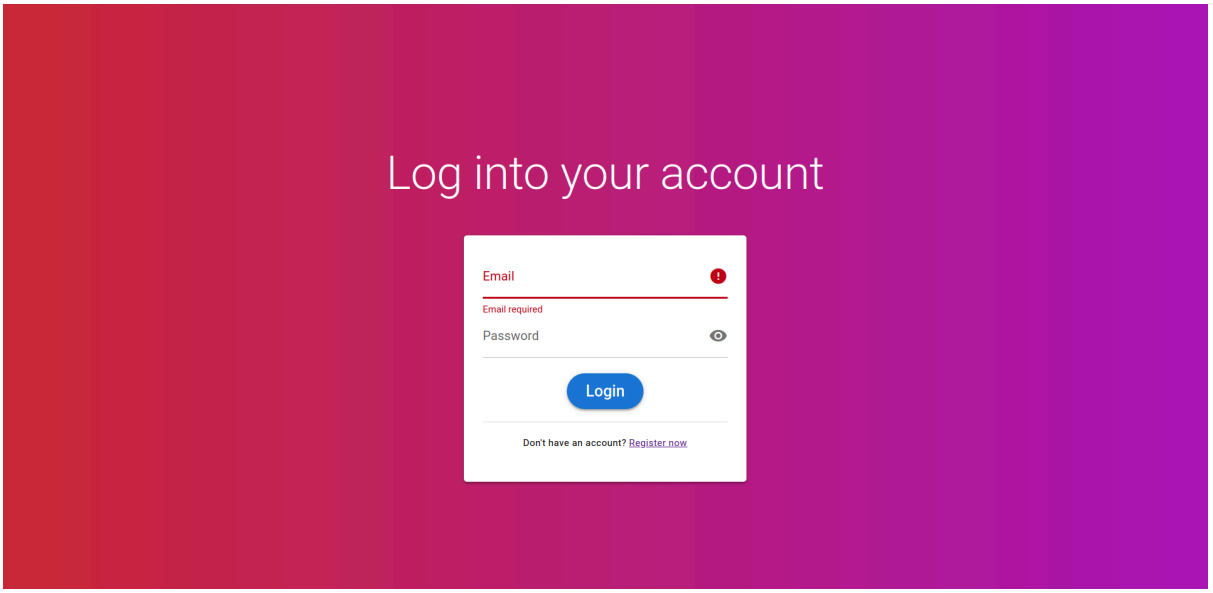
V dátovom modeli sme vykonali nasledovné zmeny oproti prvému odovzdaniu:

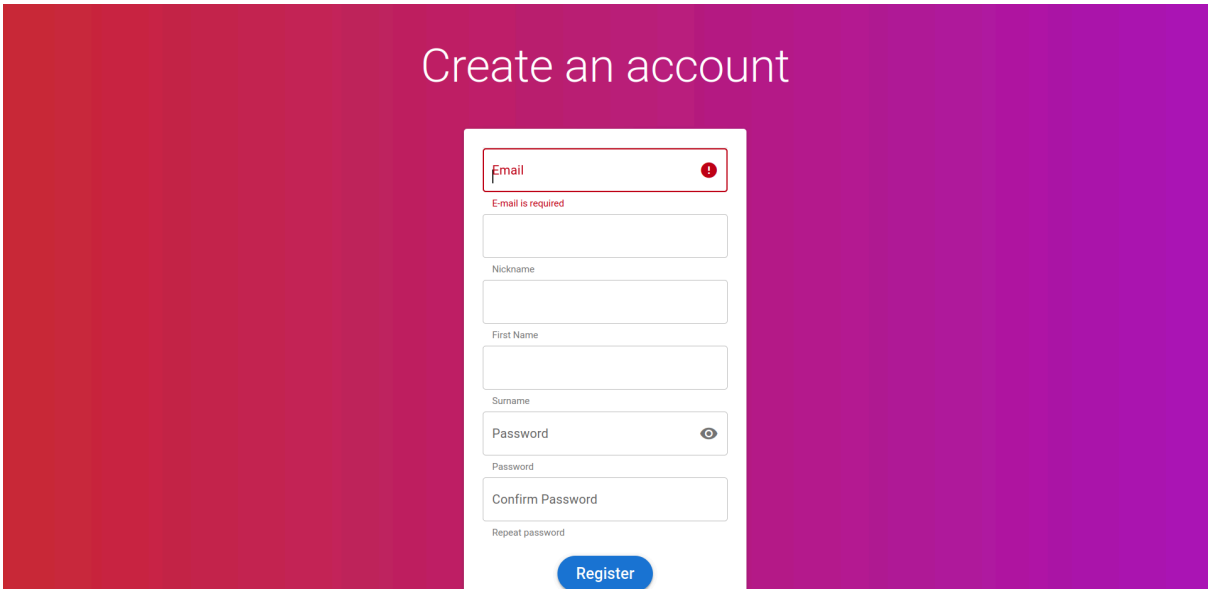
- tabuľky *adonis_schema* a *adonis_schema_versions* špecifické pre rámec AdonisJS pri prvej migrácii
- tabuľka *auth_access_tokens* pre manažment tokenov pri autentifikácii používateľa
- pribudla nová entita *updated_at* pre všetky vlastné tabuľky poskytnutá automaticky rámcom AdonisJS

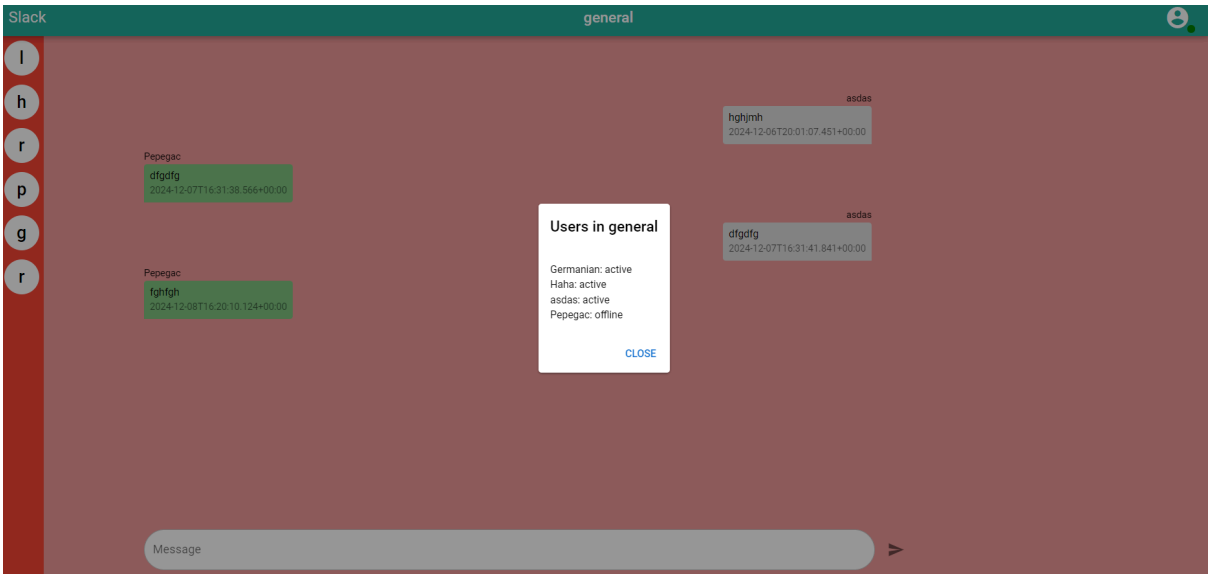
Snímky obrazoviek aplikácie


Slack
general
g
u
bomba is typing...
bomba
Ahoj
2024-12-08T17:19:13.318+00:00
domos
Cauko
2024-12-08T17:19:32.877+00:00
bomba
Pocuj
2024-12-08T17:20:16.901+00:00
bomba
@domos si tu?
2024-12-08T17:20:26.229+00:00
Message


Slack
general
g
u
bomba is typing...
bomba
Ahoj
2024-12-08T17:19:13.318+00:00
domos
Cauko
2024-12-08T17:19:32.877+00:00
bomba
Pocuj
2024-12-08T17:20:16.901+00:00
bomba
@domos si tu?
2024-12-08T17:20:26.229+00:00
bomba's Message:
tu vidime pisanu spravu
CLOSE
Message


Log into your account
Email
Email required
Password
Login
Don't have an account? Register now





Návrhové rozhodnutia

Použili sme tieto externé knižnice:

- socket.io - Túto knižnicu sme potrebovali na zabezpečenie socketov na serveri a na klientovi a vzájomnú komunikáciu medzi nimi.
- axios - Axios sme používali pri posielaní HTTP požiadaviek na server.
- dotenv - Táto knižnica nám umožnila vytvoriť env súbor do ktorého sme si mohli uložiť globálne premenné.
- pinia - Zabezpečuje store na manažovanie globálneho stavu aplikácie.

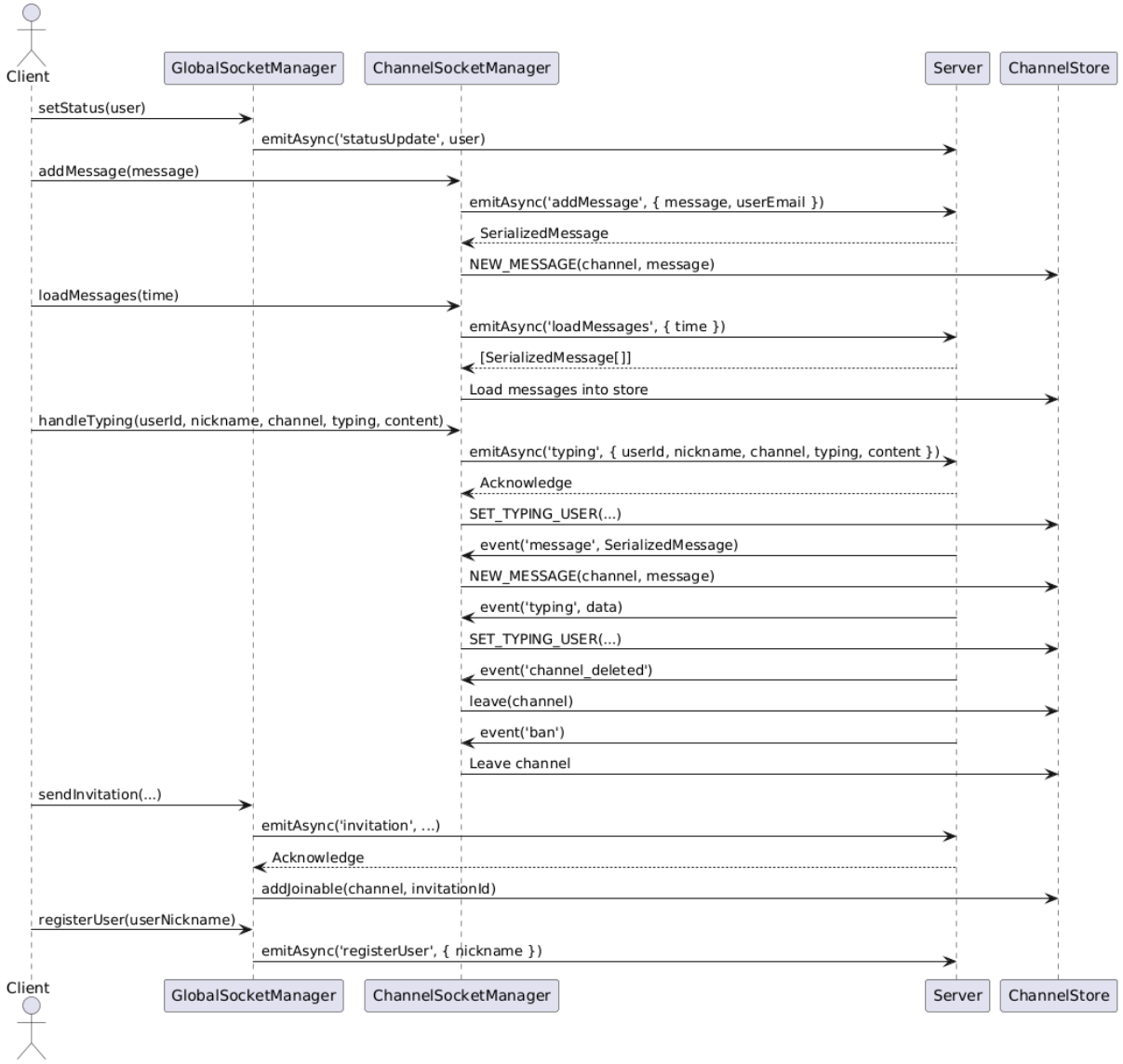
Ako šablónu použili triedy prezentované na prednáške, ktoré zabezpečujú komunikáciu so serverom prostredníctvom socketov. Na strane klienta to sú konkrétne *SocketManager*, *ChannelSocketManager*, *ChannelService* a *channelStore*. Tie sme prevzali bez väčších zmien a postupne sme do nich písali ďalšie funkcionality, ktoré sme potrebovali pre splnenie podmienok.

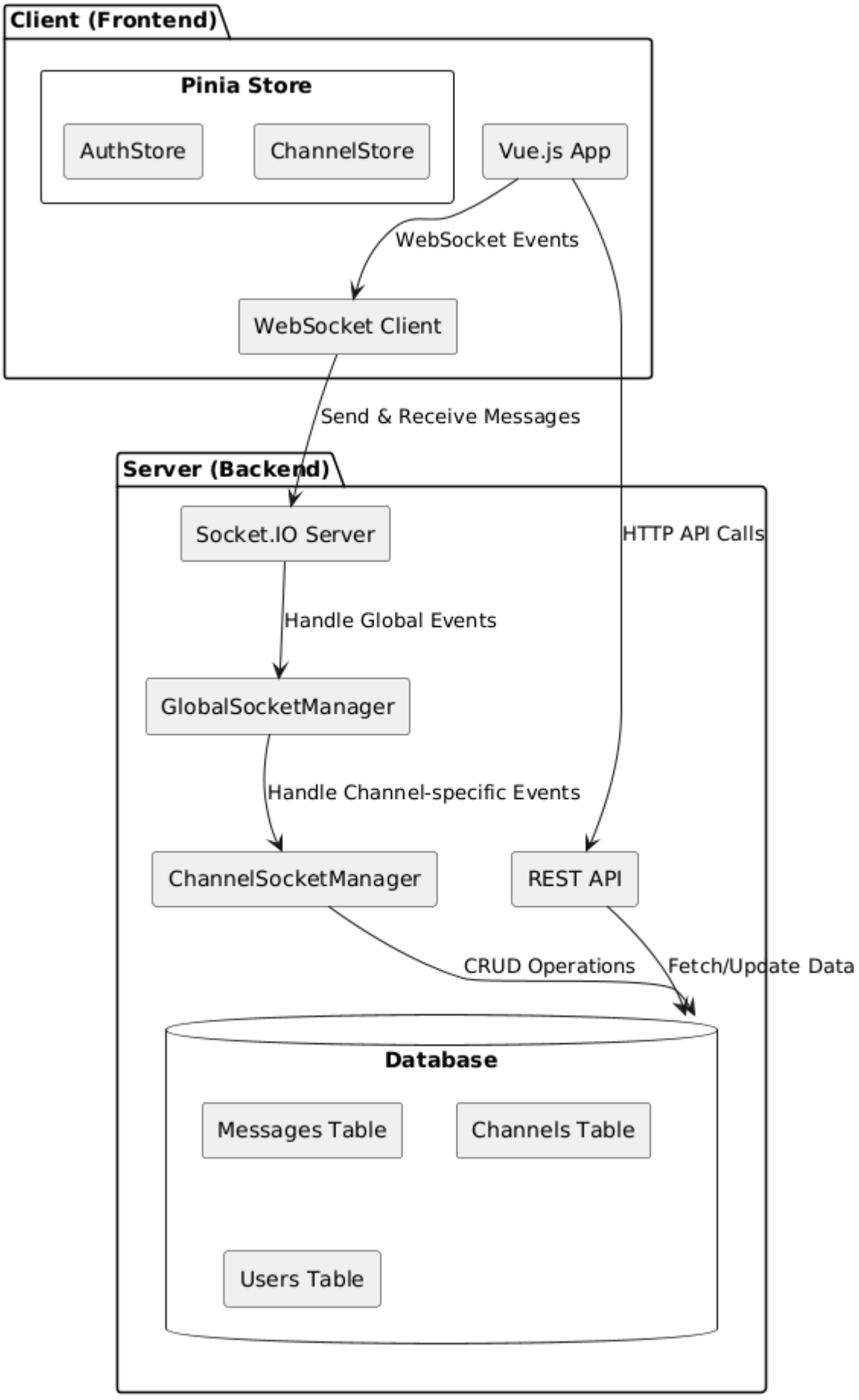
Na strane servera sme nemohli prevziať kód z prednášok, keďže sme nemohli nainštalovať AdonisJS 5. Tým pádom sme spravili vlastné manažovanie socketov pomocou triedy *ChannelManager* a globálneho namespace-u v súbore ws.ts. ChannelManager zabezpečuje namespace pre každý kanál uložený v databáze, alebo pridaný dynamicky používateľom. Ten potom prijíma a odosiela príkazy, ktoré súvisia s daným kanálom, ako správy, bany alebo či niekto píše. Potom máme globálny namespace, kde zaregistrujeme všetkých používateľov a následne im posielame príkazy, ktoré presahujú rovinu jedného kanálu. Takto napríklad posielame invite. Na začiatku sme taktiež implementovali url cesty na vytváranie/mazanie/načítanie kanálov, keďže táto činnosť nepotrebuje byť real-time.

Diagramy našej aplikácie

Rozhodli sme sa použiť 2 typy diagramov:

- Prvý sekvenčný preto, lebo používame real-time komunikáciu a je dobré vedieť ako spolu jednotlivé triedy interagujú.
- Druhým diagramom reprezentujeme samotnú štruktúru aplikácie a workflow našej aplikácie.




Client (Frontend)
Pinia Store
AuthStore
ChannelStore
Vue.js App
WebSocket Events
WebSocket Client
Send & Receive Messages
Server (Backend)
Socket.IO Server
HTTP API Calls
Handle Global Events
GlobalSocketManager
Handle Channel-specific Events
ChannelSocketManager
REST API
CRUD Operations
Fetch/Update Data
Database
Messages Table
Channels Table
Users Table