Matej Kotúč, Adam Šintál – cv. Streda 10:00

Semestrálny projekt VPWA 25/26

Aplikácia na textovú komunikáciu v štýle IRC (Slack)

1. Zadanie

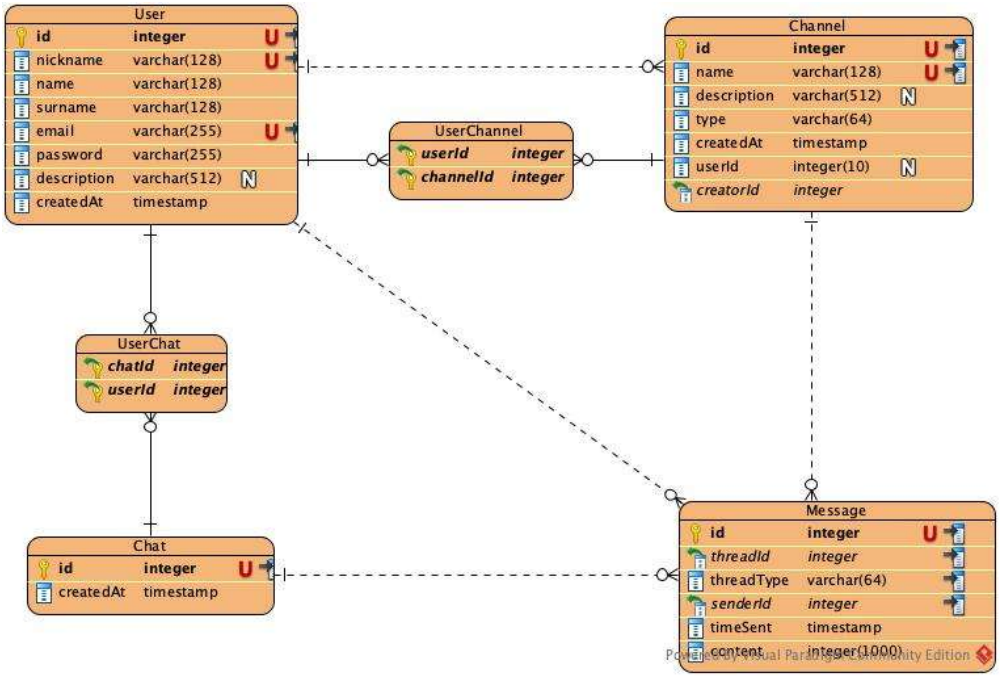
Vytvorte progresívnu webovú aplikáciu na textovú komunikáciu v štýle IRC (Slack), ktorá komplexne rieši nižšie definované prípady použitia.

1. registrácia, prihlásenie a odhlásenie používateľa
    - používateľ má meno a priezvisko, nickName a email
2. používateľ vidí zoznam kanálov, v ktorých je členom
    - pri opustení kanála, alebo trvalom vyhodení z kanála je daný kanál odobratý zo zoznamu
    - pri pozvánke do kanála je daný kanál zvýraznený a topovaný
    - v zozname môže cez používateľské rozhranie kanál vytvoriť, opustiť, a ak je správcom aj zrušiť
    - dva typy kanálov - súkromný (private channel) a verejný kanál (public channel)
    - správcom kanála je používateľ, ktorý kanál vytvoril
    - 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
    - kanál môže vytvoriť ľubovolný používateľ cez príkaz /join channelName [private]
    - do súkromného kanála môže pridávať/odoberať používateľov iba správca kanála cez príkazy /invite nickName a /revoke nickName
    - do verejného kanála sa môže pridať ľubovolný používateľ cez príkaz /join channelName (ak kanál neexistuje, automaticky sa vytvorí)
    - do verejného kanála môže člen kanála pozvať iného používateľa príkazom /invite nickName

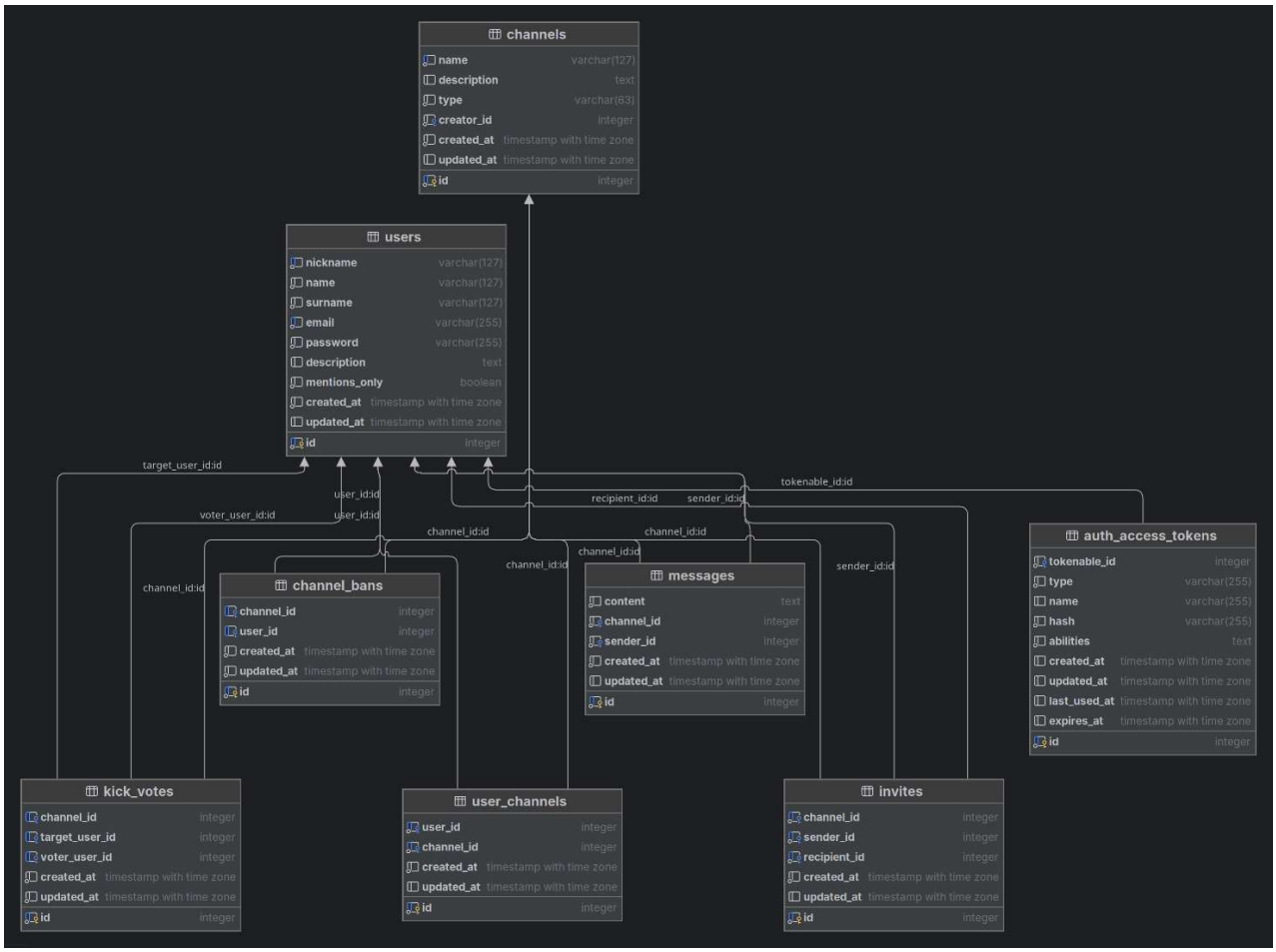
    - vo verejnom kanáli môže člen "vyhodiť" iného člena príkazom /kick nickName. ak tak spravia aspoň 3 členovia, používateľ má "trvalý" ban pre daný kanál. správca môže používateľa vyhodiť "natrvalo" kedykoľvek príkazom /kick nickName, alebo naopak "obnovit" používateľovi prístup do kanála cez príkaz /invite
    - nickName ako aj channelName sú unikátne
    - 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
    - správa je zvýraznená danému používateľovi v zozname správ
7. používateľ si môže pozrieť kompletnú históriu správ
    - efektívny inifinite scroll
8. používateľ je informovaný o každej novej správe prostredníctvom notifikácie
    - notifikácia sa vystavuje iba ak aplikácia nie je v stave "visible" (pozrite quasar docu App Visibility)
    - notifikácia obsahuje časť zo správy a odosielateľa
    - používateľ si môže nastaviť, aby mu chodili notifikácie iba pre správy, ktoré sú mu adresované
9. používateľ si môže nastaviť stav (online, DND, offline)
    - stav sa zobrazuje používateľom
    - ak je nastavený DND stav, neprichádzajú notifikácie
    - 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)
    - po kliknutí na nickName si môže pozrieť rozpísaný text v reálnom čase, predtým, ako ju odosielateľ odošle (každá zmena je viditeľná)

2. Diagram fyzického dátového modelu

1. Pôvodná verzia diagramu

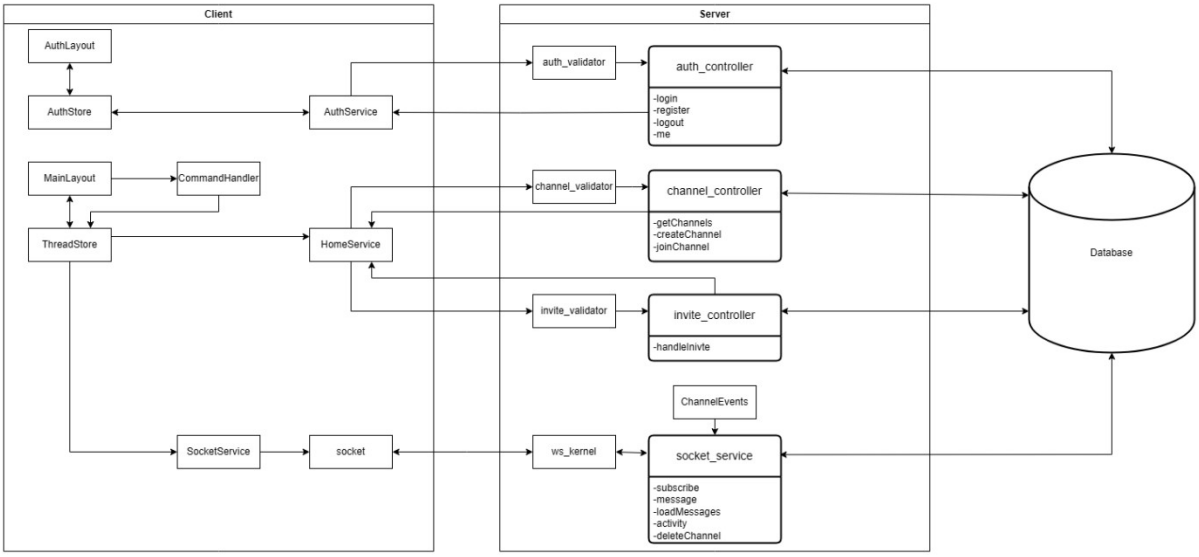


2. Vylepšená verzia diagramu



3. Zmeny
   V druhej iterácií diagramu sme pridali tabuľku channel_bans, ktorá slúži na uchovávanie zabanovaných používateľov v konkrétnom kanály.
   Taktiež bola pridaná tabuľka invites, ktorá obsahuje pozvánky pre uživateľov, čo nám umožňuje prácu s pozvánkami aj po reštartovaní serveru.
   Je dôležité spomenúť aj tabuľku auth_access_tokens, ktorú vytvára adonis automaticky a umožňuje nám prácu s autorizačnými tokenmi.
   Oproti prvej iterácií sme navyše doplnili údaje created_at a updated_at do každej tabuľky, čo nám napomáha pri práci s údajmi a prípadným identifikovaním chýb.

3. Diagram architektúry



Pri prvej interakcií s aplikáciou sa používateľ stretáva s ***AuthLayout***, ktorý poskytuje rozhranie na **prihlásenie** alebo **registráciu** do aplikácie. Tento layout čerpá dáta z ***AuthStore***, v ktorom sa ukladajú informácie o **aktuálne prihlásenom používateľovi**. Na získanie týchto dát sa používa ***AuthService***, ktorý pomocou ***http komunikácie*** posiela prihlasovacie údaje na server, kde ***auth_controller*** ovláda všetku logiku spojenú s používateľskými údajmi (ako napr. prihlásenie, registrácia, overenie používateľa na základe jeho tokenu...). Po úspešnom overení posiela server ako odpoveď prístupový token.

Po prihlásení do aplikácie sa používateľ stretáva s ***MainLayout***-om, v ktorom môže **interagovať s aplikáciou** – resp. písať do kanálov, pozývať používateľov, vytvárať nové kanály... Môže tak vykonať či už pomocou **textových príkazov** alebo **grafického rozhrania**. Informácie o dostupných kanáloch sú uschované v ***ThreadStore***. *Thread store* používa kombináciu **http requestov** (pomocou ***HomeService***) a **websocket komunikácie**. Websocket komunikácia je sprostredkovaná cez ***SocketService*** zo strany klienta, zo strany serveru počuva ***ws_kernel***, ktorý jednotlivé správy preposiela do ***socket_service***. *Socket_service* využiva triedu ***ChannelEvents***. Táto trieda umožňuje socket clientovi pripojenie do

konkrétneho kanálu (funkcia ***subscribe***), vďaka čomu môže klient prijímať **všetku komunikáciu** určenú **odoberateľom** tohto kanála (správy, aktivita...). Každý socket client môže byť odoberať **maximálne jeden kanál zároveň** – kanál ktorý má **otvorený na obrazovke**. Správy sa **načítavajú dynamicky** podľa otvoreného kanálu. Taktiež je dôležité podotknúť, že každá vytvorená komunikácia medzi serverom a klientom je **verifikovaná pomocou prístupového tokenu**.

4. Návrhové rozhodnutia

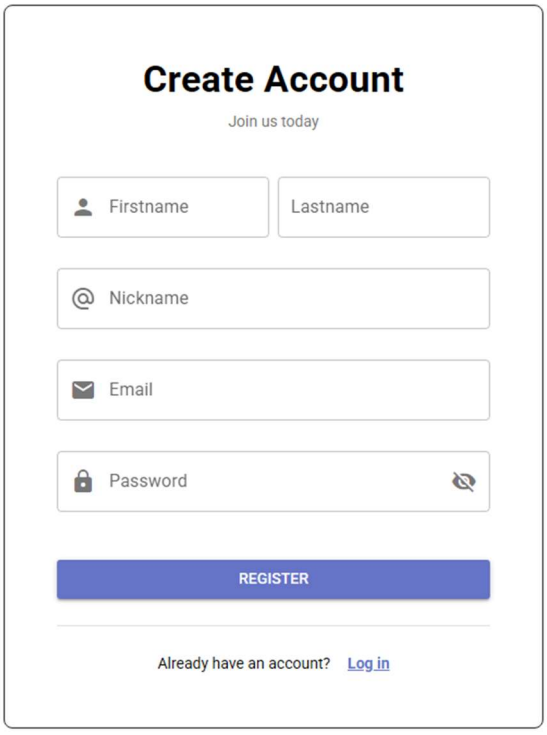
1. Frontend
   - Quasar – tučný rámec nad Vuejs, poskytuje veľké množstvo UI komponentov, má vbudovanú podporu na štýlovanie elementov a umožňuje jednoduchú implementáciu PWA.
   - Axios – axios sme použili kvôli uľahčeniu práce s http požiadavkami, taktiež podporuje interceptors, ktoré nám umožnili vkladanie autentifikačného tokenu ku každej požiadavke.
   - Pinia store – umožňuje zdieľať dáta naprieč celou aplikáciou, konkrétne sme to využili na ukladanie údajov o používateľovi alebo informácií o jednotlivých kanáloch.

2. Backend
   - AdonisJS 6 – rozsiahly rámec poskytujúci služby biznis logiky. Obsahuje vstavaný ORM (modely, migrácie...), možnosť použitia cotrollerov pre REST komunikáciu, podpora middleware, session-based auth.
   - Lucid ORM s PostgreSQL imlemetáciou – práca s databázou, jednoduchá definícia a práca s modelmi a migráciami vďaka Lucid.
   - Luxon – knižnica umožňujúca ľahšie pracovať s dátumami a časmi, jednoduchšie formátovanie a podpora časových zón.
   - Vine – knižnica používaná na validáciu údajov, zabezpečuje že dáta s ktorými pracujeme sú v správnom formáte.

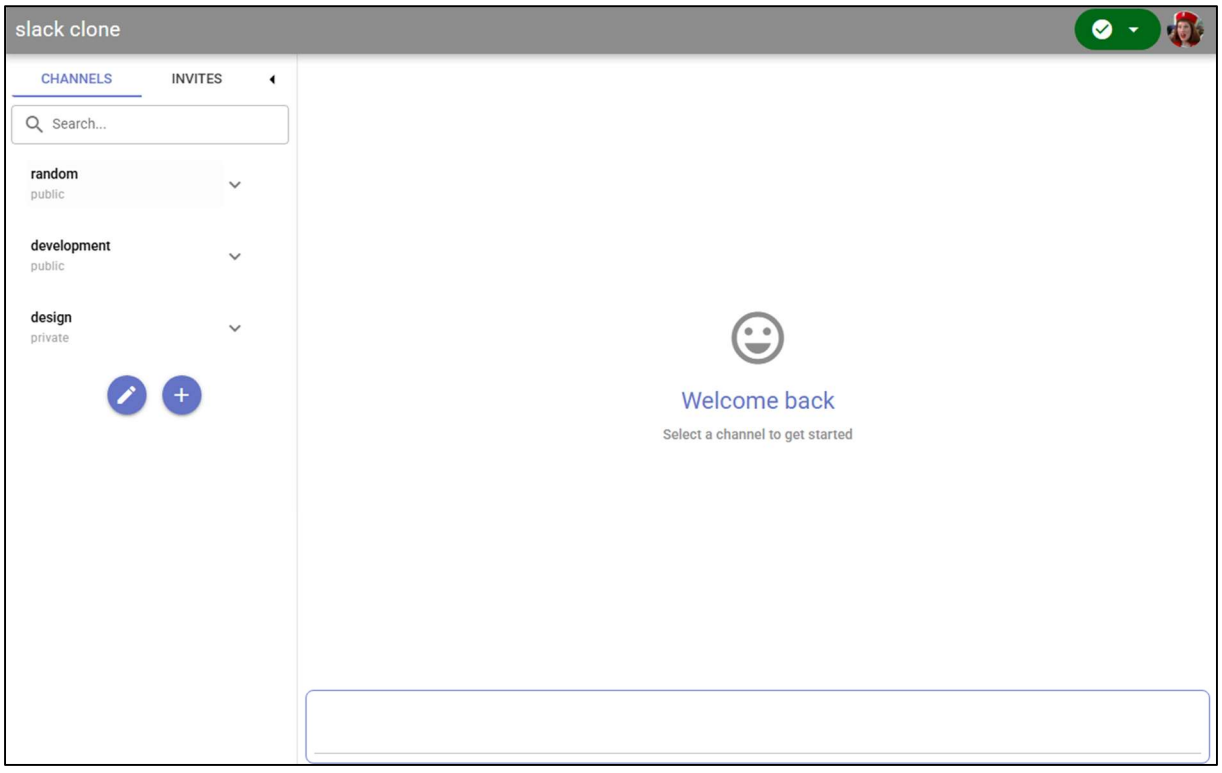
3. Spoločné
   - Socket.io – knižnica umožňujúca jednoduchú a efektívnu prácu s websocketmi, podpora používania rooms čo sa nám veľmi hodilo pre posielanie správ a notifikácií do konkrétnych kanálov. Taktiež umožňuje používať io.use, čo slúži ako middleware pre overenie tokenu.

5. Snímky obrazoviek

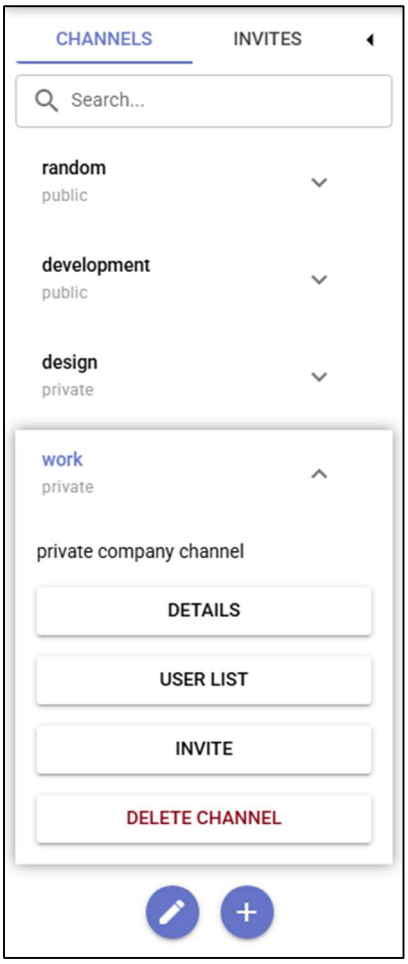
- Registrácia



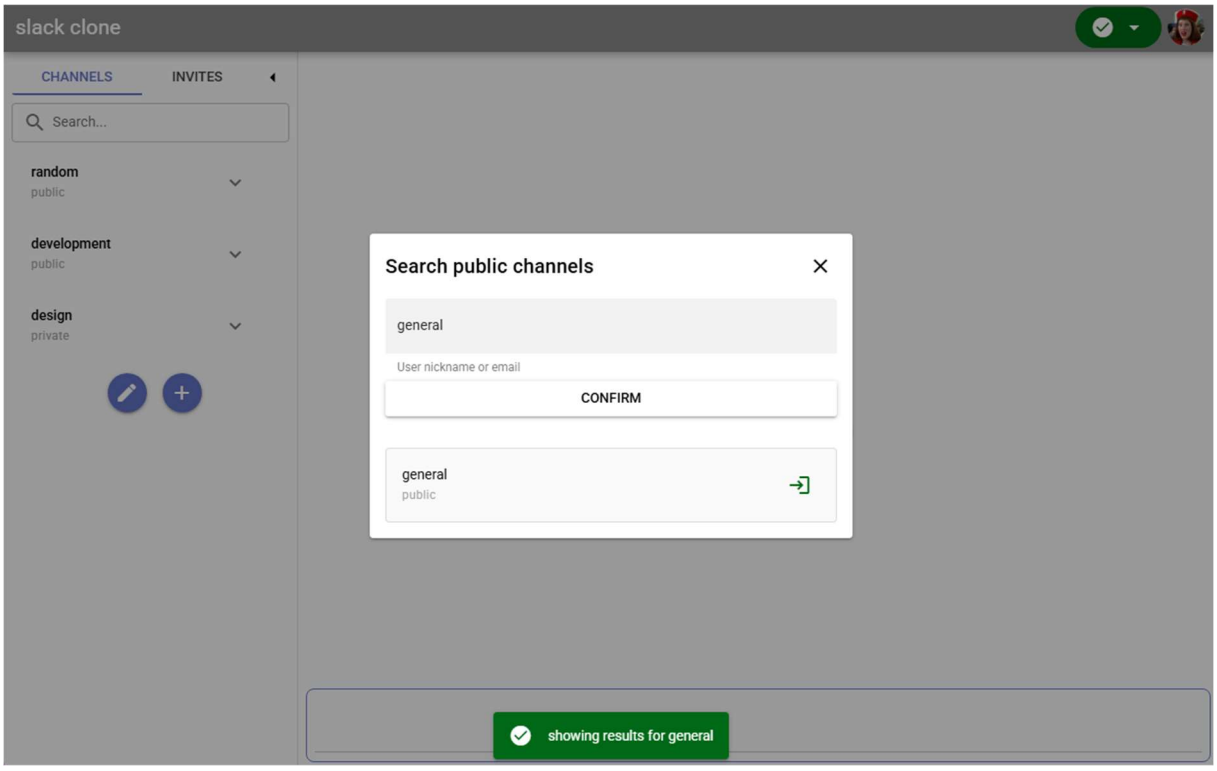
- Základný layout



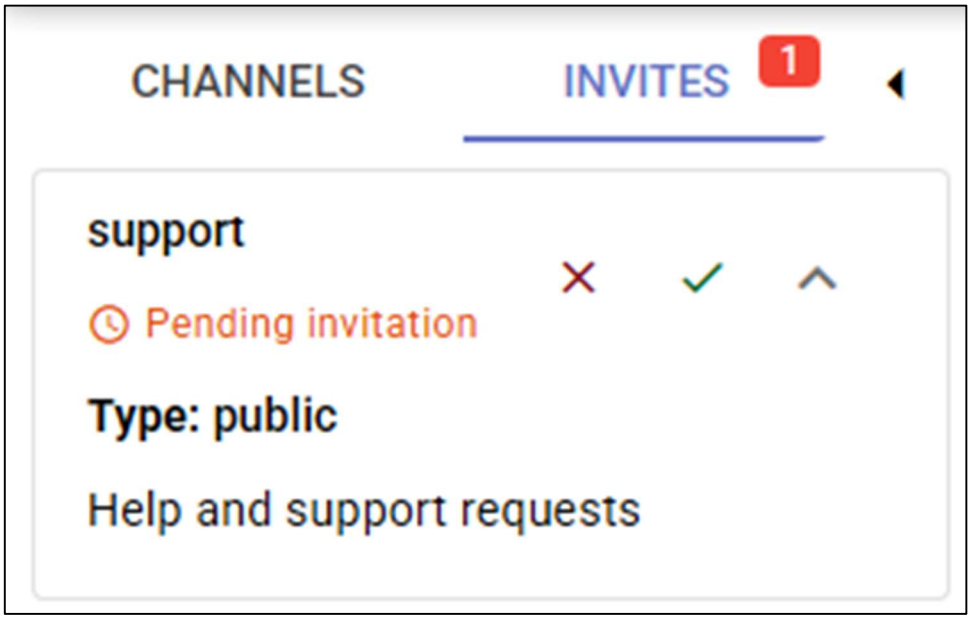
- Bočný panel



- Vyhľadávanie verejných kanálov



- Pozvánky



- Panel so správami