Dokumentacja Inżynierii Wymagań

Spis treści

1. Przedstawienie projektu

Cel Projektu

Stworzenie funkcjonalnego terminala SSH dostępnego w przeglądarce internetowej wraz z udogodnieniami pracy. Aplikacja internetowa ma zapewnić środowisko sprzyjające pracy zespołowej bez konieczności instalacji dodatkowego oprogramowania.

Przedstawienie zespołu oraz kompetencji jego członków

Kompetencje	Krzysztof Stefański	Mateusz Setkowicz
Programowanie Python	**Posiada**	**Posiada**
Znajomość Django	**Posiada**	**Nie posiada**
Programowanie JavaScript	**Posiada**	**Posiada - podstawy**
Znajomość HTML + CSS	**Posiada**	**Posiada - podstawy**
Znajomość Bootstrap5	**Posiada**	**Posiada**
Znajomość websocketów	**Posiada - podstawy**	**Posiada**
Tworzenie dokumentacji	**Posiada**	**Nie posiada**
Koordynowanie pracy zespołu	**Nie posiada**	**Posiada**
Znajomość UML	**Nie posiada**	**Nie posiada**
Testowanie oprogramowania	**Nie posiada**	**Nie posiada**

Wykorzystywane narzędzia, frameworki i języki

- Django + compontents
    - Django ORM

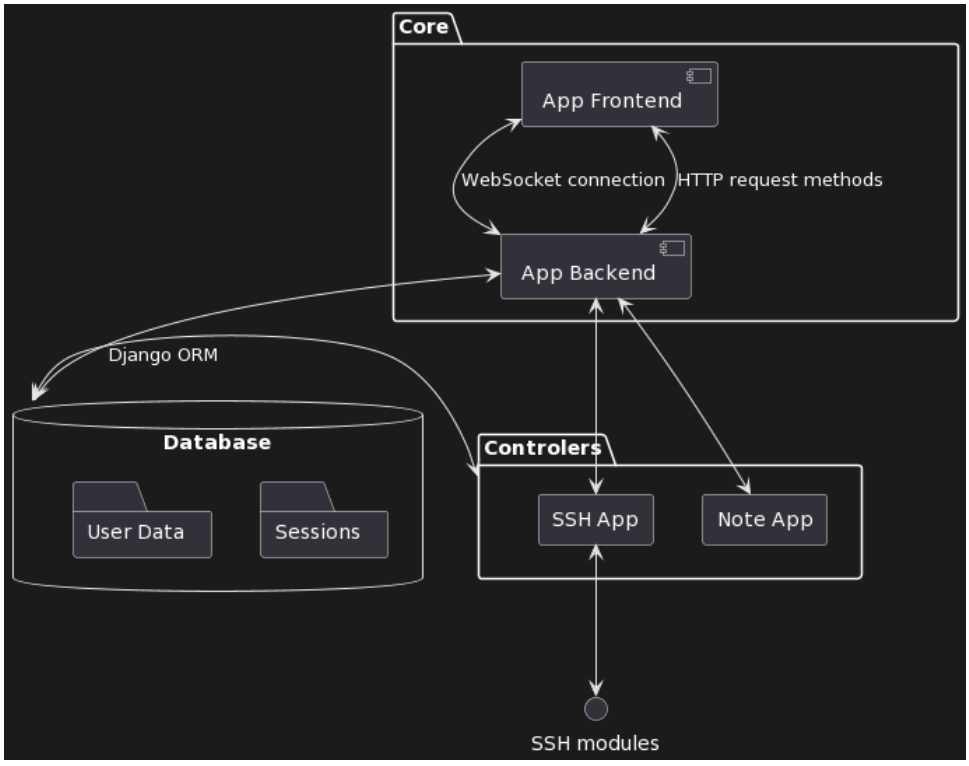
    - Django Bootstrap 5
- Websockets
- Python
- JavaScript
- HTML + CSS
- Bootstrap5

Uzasadnienie:

Zdecydowaliśmy się na wykorzystanie Pythona z racji na webową naturę naszej aplikacji. Język ten wspiera wiele frameworków odpowiadających za budowanie aplikacji webowych takich jak Django lub Flask. Z dwóch wspomnianych rozwiązań wybraliśmy Django, z racji na stosunkowo ograniczony czas wykonania projektu oraz ograniczone zasoby ludzkie. Zapewnia on wiele więcej zaimplementowanych rozwiązań i funkcjonalności, które w przypadku Flaska należy implementować samemu co jest czasochłonne i w przypadku naszego projektu niepotrzebne. Django oferuje również szeroki zakres modułów zapewniających skalowalność i bezpieczeństwo aplikacji. W naszej aplikacji skorzystamy na pewno z Django ORM. Odrzuciliśmy kwestię pisania bazy danych w czystym SQL, ponieważ wymagana przez nas struktura bazy i jej użycie nie powinna nieść ze sobą problemów z wydajnością co wymagałoby szczególnej ręcznej optymalizacji. Dodatkowo ponownie istotnym czynnikiem był czas. Nie wspominając, że w wielu wersjach przedprodukcyjnych bazy danych projektuje się korzystając z metodyki ORM.

2. Ogólna struktura Aplikacji:

Aplikacja składa się z dwóch kluczowych warstw: frontendu (interfejsu użytkownika) oraz backendu (serwera).



App Frontend:

- Odpowiada za interakcję z użytkownikiem i prezentację interfejsu graficznego.

- Współpracuje z Backend poprzez protokół WebSocket do przekazywania danych w czasie rzeczywistym.
- Wykorzystuje metody zapytań HTTP (np. GET, POST, PUT, PATCH, DELETE) do komunikacji z backendem w celu pobierania i wysyłania danych.
- Odpowiada za generowania interfejsu graficznego na podstawie danych przesłanych przez backend

App Backend:

- Odpowiada za przetwarzanie żądań użytkownika, współdzielenie zasobów między użytkownikami oraz kontrolę ilości dostępnych zasobów dla użytkowników.
- Współpracuje z modułami "SSH App" i "Note App" w celu obsługi funkcji związanych z sesjami SSH i notatkami.
- Komunikuje się z "Baza Danych" przy użyciu Django ORM (Object-Relational Mapping) do przechowywania i pobierania danych.
- Ustanawia połączenie poprzez WebSocket z frontendem w celu obsługi komunikacji w czasie rzeczywistym.

Controlers:

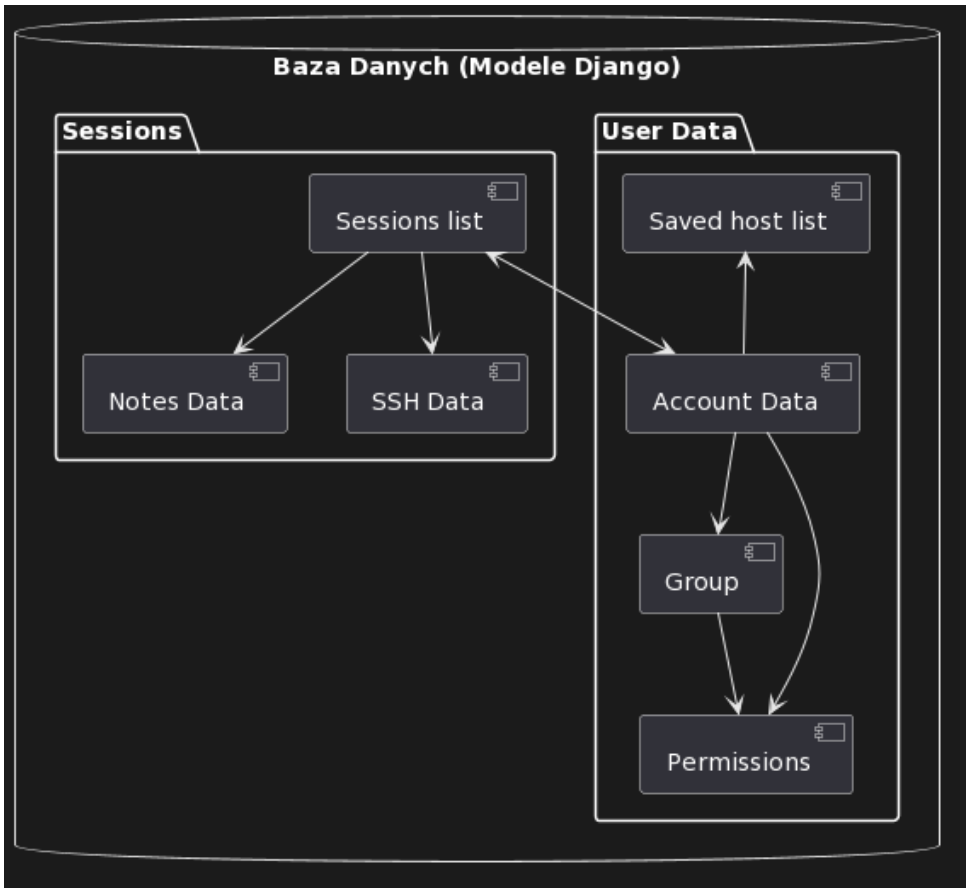
- **[SSH Module]:** Obsługuje operacje związane z sesjami SSH, takie jak tworzenie, zamykanie sesji, przetwarzanie komend, streaming danych w czasie rzeczywistym, łączenie się z zdalnymi urządzeniami oraz proces uwierzytelniania. W celu realizacji tych usług korzysta z modułu ssh (paramico)
- **[Note Module]:** Zarządza operacjami związanymi z notatkami, takimi jak tworzenie, edycja, usuwanie, zapisywanie notatek.

Database:

- Zawiera tabele z danymi, skategoryzowane na dwie główne grupy:
    - "Sessions" - informacje związane z aktywnymi sesjami SSH i sesjami notatek.
    - “User Data" - dane użytkowników, w tym dane uwierzytelniania, uprawnienia użytkowników oraz zapisana lista hostów ssh.
- Backend komunikuje się z bazą danych przy użyciu Django ORM do przechowywania, zapisywania i pobierania danych.

Struktury Bazy Danych:

Baza danych aplikacji, zrealizowana za pomocą modeli Django, obejmuje dwa główne obszary: dane użytkowników, dane sesji.



User Data:

- **[Account Data]**: Przechowuje dane użytkowników, takie jak nazwa użytkownika, hasz hasło, adres e-mail.
- **[Group]**: Grupuje użytkowników na podstawie wspólnych cech, uprawnień lub ograniczeń.
- **[Permissions]**: Przechowuje informacje o uprawnieniach użytkowników, takich jak dostęp do określonych funkcji aplikacji.

- **Relacje:**
    - [Account Data] jest powiązane z [Group] w sposób wielu do wielu, co pozwala przypisywać użytkowników do różnych grup.
    - [Account Data] jest również powiązane z [Permissions] w sposób wiele do wielu, co pozwala kontrolować dostęp użytkowników do różnych funkcji aplikacji.
    - [Saved host list]: Przechowuje listę zapisanych hostów (IP address/hostname sposób uwierzytelniania hasło/key) związanych z kontem użytkownika.

Sessions:

- **[Sessions list]**: Przechowuje informacje o aktywnych sesjach użytkowników. Określa kto jest połączony do danej sesji oraz określa jego uprawnienia: do udostępniania, zamykania sesji.
- **[Notes Data]**: Zawiera dane dotyczące sesji notatek.
- **[SSH Data]**: Przechowuje informacje o sesjach SSH z urządzeniami zdalnymi.
- **Relacje:**

- [Notes Data] jest powiązane z [Sessions list] w sposób jeden do wielu, co umożliwia przechowywanie informacji o wielu sesjach użytkowników związanych z konkretną notatką.
- [SSH Data] jest również powiązane z [Sessions list] w sposób jeden do wielu, co umożliwia przechowywanie informacji o wielu sesjach użytkowników związanych z konkretnym połączeniem SSH.

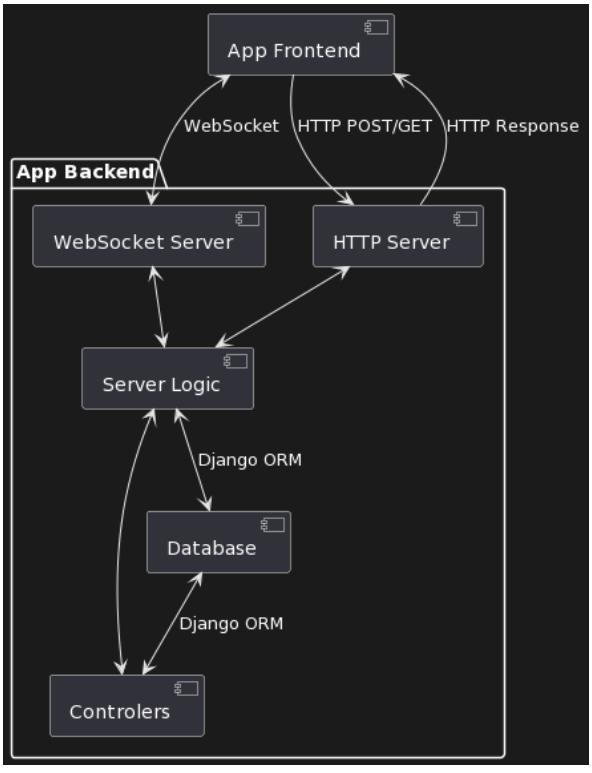
Uprawnienia użytkownika (Tabela Permissions)

Administrator strony będzie odpowiedzialny za przydzielanie uprawnień użytkownikom poprzez zarządzanie tabelą Permissions poprzez wbudowane narzędzie “Panel Administracyjny Django”. Ta funkcjonalność ma na celu kontrolowanie, co użytkownicy mogą robić w ramach swoich kont. Poniżej przedstawione są planowane uprawnienia, które zostaną zaimplementowane:

- **Uprawnienie "Note":**
  - Uprawnienie to pozwala użytkownikowi korzystać z funkcjonalności związanej z notatkami.
  - Gdy użytkownik posiada to uprawnienie, może tworzyć, edytować, usuwać oraz przeglądać notatki.
- **Uprawnienie "SSH":**
  - Uprawnienie to pozwala użytkownikowi korzystać z funkcjonalności związanej z sesjami SSH.
  - Użytkownik z uprawnieniem "SSH" może otwierać, zamykać sesje SSH, przesyłać komendy oraz korzystać z terminala zdalnego.

Funkcjonalność ta ma na celu zwiększenie kontroli nad tym, jakie czynności użytkownicy mogą wykonywać w systemie.

Diagramy przepływu danych:



HTTP POST:

- Dane uwierzytelniające, takie jak nazwa użytkownika i hasło, są przesyłane do [HTTP Server] w celu uwierzytelnienia użytkownika.

HTTP DELETE, PATCH, PUT:

- Dane do zarządzania sesji, udostępniania, pobierania podstawowych informacji o wyglądzie strony.

HTTP GET:

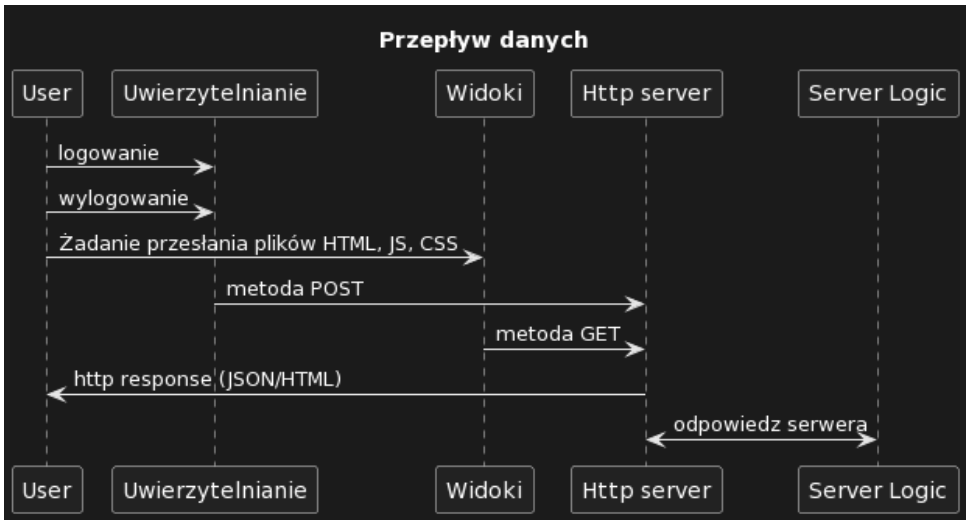
- Wysyłanie stron HTML wraz z kodem JavaScript, który służy do generowania interfejsu użytkownika.

WebSocket:

- Służy do przesyłania danych w czasie rzeczywistym w obie strony (z i do WebSocket Servera), takich jak aktualizacje interfejsu użytkownika, powiadomienia o zmianach, przesyłanie nowych danych oraz przesyłanie żądań użytkownika i odpowiedzi z modułów.
- Przesyłane dane sesji, umożliwiając generowanie interfejsu użytkownika dostosowanego do zalogowanego użytkownika.

HTTP Response:

- [HTTP Server] odpowiada na żądania HTTP z [App Frontend] za pomocą odpowiedzi HTTP.
- Odpowiedzi mogą zawierać potwierdzenia operacji, dane pobrane za pomocą GET, itp.



Przetwarzanie danych w Server Logic:

- [HTTP Server] i [WebSocket Server] przekazują dane do warstwy [Server Logic], gdzie są przetwarzane.
- Logika serwera decyduje, jakie operacje należy wykonać na podstawie otrzymanych danych i w jaki sposób obsłużyć zapytanie.

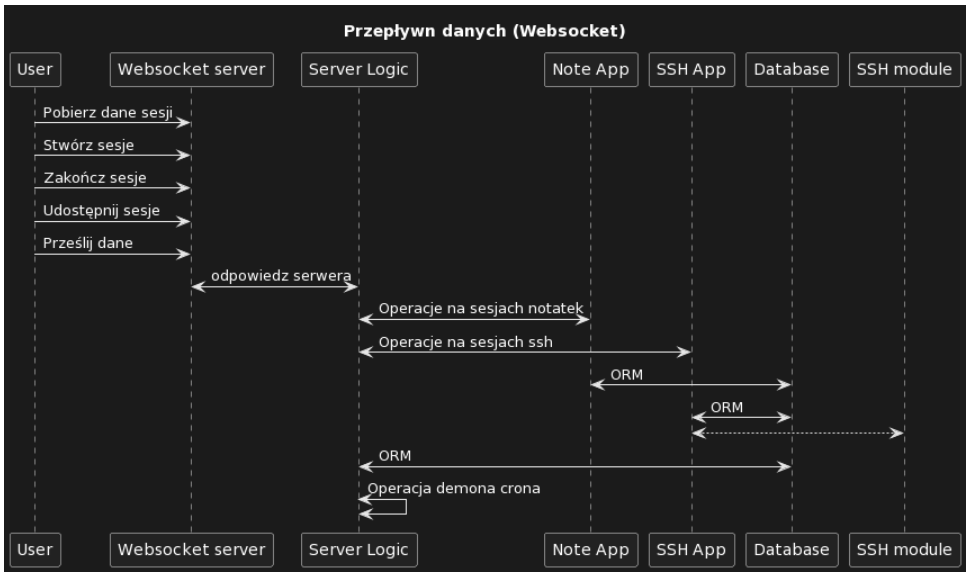
Kontrolers:

- [Server Logic] wywołuje odpowiedni kontroler (odpowiednią aplikację) w celu przetworzenia zapytania użytkownika.

- Aplikacje te realizują operacje związane z zapisem/odczytem danych z bazy danych. Mogą obejmować różnorodne operacje, takie jak zarządzanie sesjami, tworzenie, zamykanie i ich edycja

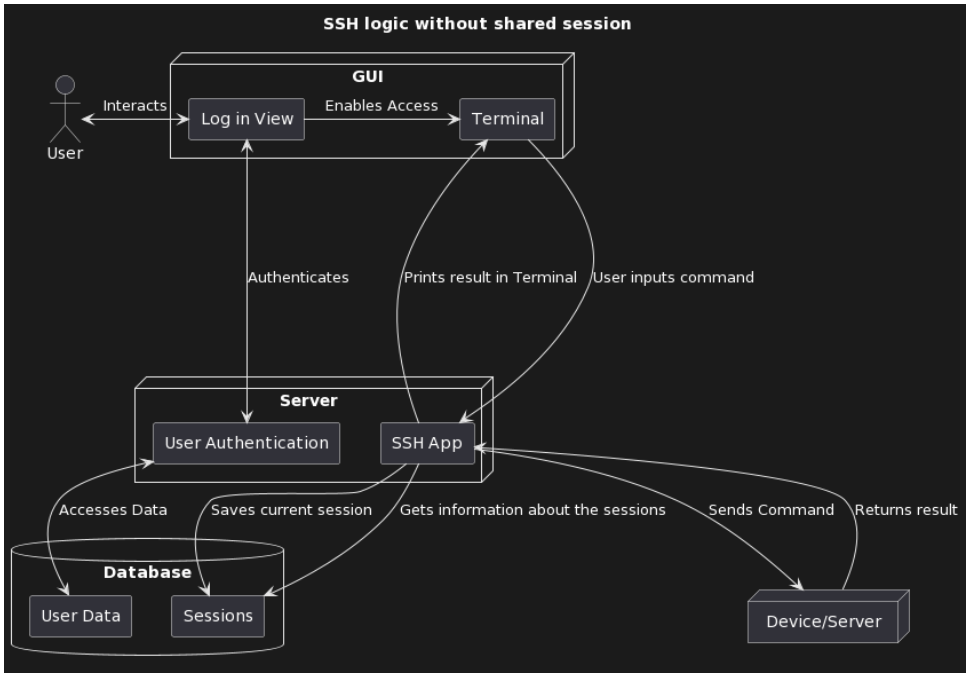
Django ORM:

- [Server Logic] korzysta z Django ORM, aby komunikować się z [Database] w celu uwierzytelniania użytkownika i autoryzacji jego dostępu do odpowiednich zasobów.
- Pozwala [Controlers] na pobieranie i zapisywanie danych związanych z sesjami użytkownika



3. Uogólnione schematy działania

Obsługa sesji SSH



Uwierzytelnianie

Istniejące konto

- Użytkownik wprowadza dane logowania
- Backend weryfikuje dane wprowadzone przez użytkownika zestawiając je z wpisami w bazie danych
- Udziela lub odmawia dostępu

Wykonywanie poleceń SSH

Nowa sesja

- Użytkownika wprowadza polecenie do terminala
- Polecenie zostaje przesłane do modułu odpowiadającego za logikę związaną z SSH
- Moduł SSH wysyła polecenie do urządzenia / serwera
- Zapisuje odpowiedź w bazie danych

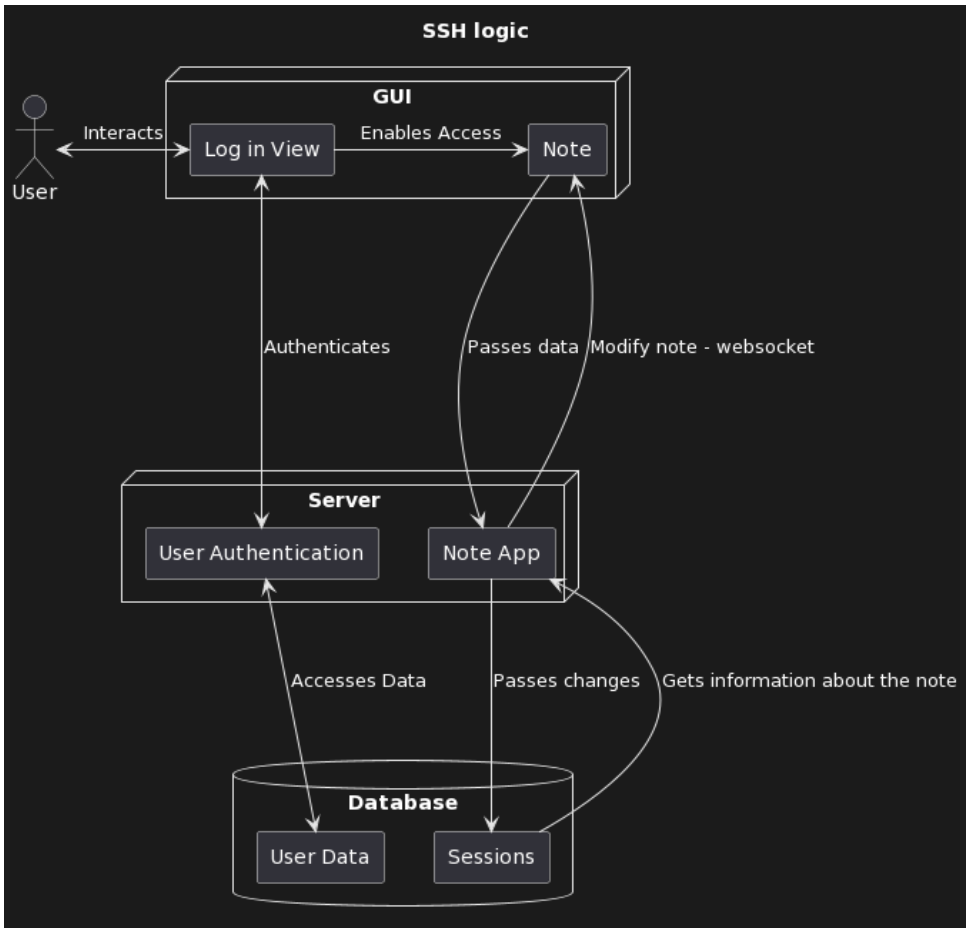
Istniejąca sesja

- SSH moduł pobiera informacje dotyczące sesji
- Terminal jest aktualizowany
- Następne kroki przebiegają identycznie jak dla nowej sesji

Współdzielona sesja

- W przypadku współdzielonej sesji [SSH App] w momencie otrzymania komendy aktualizuje terminal dla wszystkich użytkowników danego połączenia websocket
- Podobnie w przypadku otrzymania odpowiedzi od serwera

Obsługa notatek



Uwierzytelnianie

Istniejące konto

- Użytkownik wprowadza dane logowania
- Backend weryfikuje dane wprowadzone przez użytkownika zestawiając je z wpisami w bazie danych
- Udziela lub odmawia dostępu do zasobów serwera

Notatka

Nowa notatka

- Wprowadzane zmiany są zapisywane w bazie danych

Istniejąca notatka

- Moduł pobiera informacje na temat notatki
- Widok jest aktualizowany
- Następne kroki przebiegają identycznie jak dla nowej notatki

Współdzielona sesja

- Wprowadzanie zmian emituje zdarzenie przy użyciu protokołu websocket
- Dla wszystkich użytkowników danego połączenia websocket notatka jest aktualizowana

Opis komponentów fronendu:

- **Okno Główne:** obszar roboczy, gdzie użytkownicy widzą treść aktualnie otwartego terminala SSH lub notatki. Wyświetla zawartość sesji SSH lub notatki, umożliwia wprowadzanie komend w terminalu/ edycję notatek.
- **Lista Szybkiego Wyboru Sesji SSH:** Panel z listą zapisanych sesji SSH, które użytkownik może szybko wybrać do ponownego połączenia się z hostem. Umożliwiają szybkie łączenie się z zapisanymi sesjami SSH bez konieczności ponownego wprowadzania danych połączenia.
- **Zakładki Otwartych Sesji:** Wizualne reprezentacje otwartych sesji SSH, każda sesja reprezentowana jako zakładka. Umożliwiają użytkownikowi łatwe przełączanie się między otwartymi sesjami, ułatwiając zarządzanie wieloma terminalami jednocześnie.
- **Ikona Profilu konta użytkownika:** Umożliwia szybki dostęp do kluczowych funkcji bez konieczności przeszukiwania menu. Zawiera przycisk lub opcje, która umożliwia użytkownikowi bezpieczne wylogowanie się z aplikacji.

Wymagania funkcjonalne:

Terminal SSH:

- Umożliwia użytkownikom otwieranie wielu terminali i sesji wykonywanych po stronie serwera.
- Sesje powinny być współdzielone między użytkownikami, co oznacza widoczność tego samego outputu dla wielu użytkowników jednocześnie. Współdzielenie sesji oznacza, że użytkownicy mogą jednocześnie pracować w tej samej sesji SSH, widząc aktualizacje w czasie rzeczywistym.

Notatnik:

- Zapewnia tworzenie usuwanie notatek okiem oraz podstawowe funkcje edycji ich: pogrubienie, kursywa, zmiana wielkości czcionki, dodawanie list, zmiana koloru czcionki
- Notatnik można udostępniać - umożliwia wielu użytkownikom jednoczesną pracę nad jedną notatką, co pozwala na współpracę i dzielenie się informacjami.

Zarzadzanie platformą:

- Umożliwia użytkownikom logowanie się do aplikacji za pomocą unikalnych identyfikatorów.
- Tworzenie kont zajmuję się administrator aplikacji przez “Panel Administracyjny Django”
- Przechowywanie danych użytkownika: Zapisuje dane użytkownika, takie jak preferencje, notatki, sesje SSH.
- Dostęp do funkcjonalności notatek/ssh można ustawić dla każdego konta osobno

Lista z szybkim wyborem:

- Oferuje funkcjonalność szybkiego łączenia się z urządzeniem na podstawie wcześniej zapisanych danych - umożliwia użytkownikom szybkie łączenie się z wcześniej skonfigurowanymi sesjami terminala SSH lub notatkami.
- Każde konto posiada swoją osobną listę zapisanych sesji SSH

Wymagania niefunkcjonalne:

Obsługa równoczesnych sesji

- Aplikacja powinna obsługiwać efektywnie obsługiwać co najmniej 20 równoczesnych sesji SSH w umiarkowanym czasie odpowiedzi. Czas odpowiedzi powinien być akceptowalny dla użytkownika, nie będąc jednocześnie zbyt długi [<0.5s].

Dostępność:

- System ma być dostępny o ile nie są przekroczona maksymalne limity sesji notatek oraz sesji ssh.
- Domyślny limit to 100 sesji ssh i 100 sesji notatek
- Każda próba utworzenia nowej sesji wykraczająca poza zadaną pule będzie odrzucana
- Limity aplikacji będzie można zmienić w pliku settings.py

Bezpieczeństwo:

- Aplikacja musi zagwarantować bezpieczny transfer danych, szczególnie podczas operacji, takich jak logowanie i współdzielenie poufnych informacji.
- Hasła do kont użytkowników muszą być przechowywane w sposób bezpieczny

Prostota obsługi:

- Interfejs użytkownika powinien być intuicyjny i łatwy w obsłudze, nawet dla nowych użytkowników. Prostota obsługi ma kluczowe znaczenie dla przyjaznego środowiska pracy, umożliwiającego użytkownikom skorzystanie z funkcji bez zbędnych trudności.

4. Ograniczenia aplikacji

Pytania sformułowane w celu opracowania założeń dotyczących ograniczeń aplikacji:

Pytanie	**Odpowiedź**	**Uwagi**
Kiedy współdzielona sesja ssh ma się zakończyć? - Kiedy wpis zostanie usunięty z bazy?	Kiedy żaden użytkownik nie będzie miał otwartego okna z daną sesją.	Sesja automatycznie się zamyka gdy żadne użytkownik z niej nie korzytsa
Co, jeśli użytkownik zamknie okno z sesją ssh, którą sam udostępnił?	Wpis zostaje usunięty z bazy danych i użytkownik nie ma już dostępu do tej sesji.	Sesja dalej zamyka się u wszystkich
Czy wpisy o sesjach ssh będą przechowywane w bazie w nieskończoność?	Tak, ale sama sesja zostanie zakończona zamknięta jeśli użytkownik zamknie przeglądarkę.	Użytkownik będzie mógł odnowić nieaktywną sesje otwierając przeglądarkę ponownie
Czy wspomniane wyżej ograniczenia dotyczą notatek?	Tak	Z tą różnicą, że sesja notatek jest wiecznie aktywna, dopóki nie zamkniemy konta związanej z daną notatką
Czy użytkownik ma ograniczenie co do liczby jednocześnie otwartych sesji?	Tak, jest to 10 okien sesji ssh i 10 okien sesji notatek	Liczba może zostać dostosowana na etapie testów wydajności.

Jaka jest maksymalna ilość znaków, którą może przechowywać każda z sesji ssh/note?	Brak ograniczenia	Liczba może zostać dostosowana na etapie testów wydajności.
Czy serwer może odrzucić stworzenie sesji po przekroczeniu limitu zasobów?	W settings.py będzie możliwosc konfiguracji maksymalnej liczby sesji ssh oraz sesji notatek. Po przekorczeniu tego progu próba stworzenia sesji jest odrzucana. Domyśla ilość sesji osobno dla ssh i notatek dla całego serwer to 100	Liczba może zostać dostosowana na etapie testów wydajności.