Kalambury

1. Wchodzimy na stronę logowania:
   1. Pomysł
   2. Serwer -> Payara
   3. Baza danych
      1. Konfuguracja na serwerze Payara (screen z konsoli Payara – connection pool)
      2. Konfiguracja na backendzie (persistence.xml)
      3. Schemat bazy danych
2. Wpisać user i password i kliknąć enter
   1. Przesłanie kredencjałów do backendu
   2. Jeśli user istnieje w bazie danych, to sprawdź hasło:
      1. Pobierz hash i sól z bazy danych
      2. Podane hasło zahaszuj z pobraną z bazy solą
      3. Porównaj hasze
   3. Jeśli user nie istnieje, to stwórz nowe konto
      1. Wygeneruj losową sól dla danego użytkownika
      2. Zahaszuj hasło z dodatkiem wygenerowanej soli (PBKDF2)
         1. Dlaczego akurat ten algorytm? (<https://www.baeldung.com/java-password-hashing>) Bo jest jest silniejszy i wolniejszy od np. SHA-512, pozwala manewrować trudnością.
         2. Dlaczego sól? *(By introducing this randomness, we increase the hash's*entropy*, and we protect our database against pre-compiled lists of hashes known as*rainbow tables*.)*
      3. Zapisz hasz i sól w bazie danych
   4. Otworzyć pgAdmin i pokazać że w bazie danych siedzą hashe (
   5. Zalogowanemu userowi wygeneruj token JWT i prześlij go na frontend.
      1. Czym są JWT?
      2. Mechanizm podpisywania tokenu
      3. Jak działa weryfikacja tokenu (<https://github.com/auth0/java-jwt> tu jest instrukcja jak to działa w Javie w bibliotece, której używamy)
   6. Token JWT zapisz w Local Storage przeglądarki
      1. Pokazać devtools->applications na dwóch przeglądarkach, że zapisują siętokeny
3. Narysować coś na płótnie
   1. Websocket -> full duplex, asynchroniczny
   2. Na otwarcie połączenia websocket frontend wybiera z Local Storage swój token i przesyła do backend w celu weryfikacji. Sprawia to, że nikt kto nie ma tokena nie podszyje się pod użytkownika.
4. Pokazać, że działa resize (gdy użytkownik o mniejszym ekranie coś wyśle, to innym ten obraz się przeskaluje) (teraz czy później? W ogóle to pokazywać?)