A screenshot of a computer

Description automatically generated

**PROJEKT INŻYNIERSKI**

„Internetowa platforma crowdfundingowa”

**Maciej Maciejewski**

**Nr albumu 290398**

**Kierunek:** Informatyka

**Specjalność:** Grafika komputerowa i oprogramowanie

**PROWADZĄCY PRACĘ**

**Dr inż. Artur Pasierbek**

**KATEDRA Algorytmiki i Oprogramowania**

**Wydział Automatyki, Elektroniki i Informatyki**

**GLIWICE Rok 2023/24**

**Spis treści**

Rozdział 1 Wstęp

* 1. Wprowadzenie do tematu projektu
  2. Cel pracy
  3. Charakterystyka zawartości rozdziałów

Rozdział 2 Analiza tematu

2.1 Sformułowanie problemu

2.2 Opis istniejących rozwiązań

Rozdział 3 Wymagania i narzędzia

3.1 Wymagania funkcjonalne

3.2 Wymagania niefunkcjonalne

3.3 Przypadki użycia

3.4 Opis narzędzi

3.4.1 Języki programowania

3.4.2 Frameworki

3.4.3 Baza danych

3.4.4 Środowisko i Narzędzia

3.4.5 Deployment

3.5 Metodyka pracy

Rozdział 4 Specyfikacja zewnętrzna

4.1 Wymagania sprzętowe i programowe

4.1.1 Lokalnie

4.1.2 Docker

4.2 Sposób instalacji

4.3 Sposób uruchomienia

4.4 Realizacja głównych przypadków użycia

4.4.1 Rejestracja i logowanie

4.4.2 Tworzenie nowego projektu jako użytkownik

4.4.3 Wsparcie wybranego projektu

4.5 Administracja serwisu

4.6 Bezpieczeństwo aplikacji

4.6.1 JWT

4.6.2 Mechanizm ról użytkownika

4.6.3 CanActivate

4.6.4 404

Rozdział 5 Specyfikacja wewnętrzna

5.1 Architektura serwisu

5.1.1 Architektura po stronie backendu

5.1.2 Architektura po stronie frontendu

5.2 Schemat bazy danych

5.3 Wykorzystane biblioteki

5.3.1 Backend

5.3.2 Frontend

5.4 Istotne fragmenty kodu

5.4.1 Backend

5.4.2 Frontend

5.5 Kluczowe algorytmy

Rozdział 6 Weryfikacji i walidacja

6.1 Sposób testowania

6.2 Zakres testowania

6.3 Przypadki testowe

6.4 Wykryte i usunięte błędy

Rozdział 7 Podsumowanie i wnioski

Bibliografia

Spis skrótów i symboli

Źródła

Lista dodatkowych plików, uzupełniających tekst pracy

Spis rysunków

Spis tablic

**1.Wstęp**

**1.1 Wprowadzenie do Tematu Projektu**

W dobie cyfrowej transformacji oraz globalizacji, której początek datuje się na moment upublicznienia Internetu i która nieprzerwanie trwa do dzisiaj, znalezienie ludzi o podobnych zainteresowaniach lub problemach stało się zadaniem niezwykle łatwym. Ta nowa rzeczywistość cyfrowa umożliwia łączenie osób z różnych zakątków świata, tworząc społeczności oparte na wspólnych pasjach, celach czy potrzebach. Niniejsza praca inżynierska koncentruje się na wykorzystaniu tego fenomenu w kontekście internetowej platformy crowdfundingowej.

Tradycyjnie inwestorem w wielu projektach jest jeden lub kilka podmiotów które już posiadają znaczący kapitał. Największą wadą takiego modelu finansowania jest to że inwestycja powinna się zwrócić inwestorom przez co twórca lub zespół muszą zwracać durzą uwagę na to ile są w stanie zarobić na projekcie, co może prowadzić do zmniejszenia jakości produktu lub agresywnego modelu monetyzacji, przykładowo zablokowanie niektórych elementów przez subskrypcją lub dodatkową opłatą, co potrafi zrazić wiele osób które na początku były podekscytowane projektem.

Dla osób dla których zrobienie dobrego produktu jest ważniejsze niż wzbogacenie się na nim crowdfunding jest idealnym rozwiązaniem. Crowdfunding, czyli finansowanie społecznościowe, oznacza że projekt nie ma jednego lub kilku znaczących inwestorów ale ma ich cały tłum (eng. crowd) który finansuje (eng. fund) projekt nie dużymi kwotami ale jak mówi przysłowie grosz do grosza, a będzie kokosza. Oznacza to że osoby które zainwestowały są zainteresowane pomysłem przedstawionym przez twórcę. Nie interesuje ich zwrot na inwestycji tylko szczerze podoba im się produkt.

Dzięki temu mechanizmowi, innowacyjne pomysły, które mogą nie znaleźć uznania w tradycyjnych modelach finansowania, mają szansę na realizację. O ile sam pomysł trafi do dostatecznie dużo zainteresowanych osób.

**1.2 Cel Pracy**

Celem niniejszej pracy jest zaprojektowanie funkcjonalnej i intuicyjnej platformy crowdfundingowej. Głównym zadaniem jest stworzenie przestrzeni, w której twórcy i osoby kreatywne mogą łatwo znaleźć wsparcie od osób o podobnych zainteresowaniach lub borykających się z podobnymi wyzwaniami. Platforma ta ma za zadanie umożliwić łączenie różnorodnych projektów z potencjalnymi donatorami i inwestorami, tworząc wspólnotę wspierającą się nawzajem w realizacji kreatywnych i innowacyjnych pomysłów. Dodatkowym kluczowym aspektem projektu jest opracowanie przejrzystego, zrozumiałego i intuicyjnego interfejsu użytkownika, co ma kluczowe znaczenie dla zapewnienia wygodnego i efektywnego korzystania z serwisu. Równie istotne jest zapewnienie wysokiego poziomu bezpieczeństwa, zarówno od strony frontendowej, jak i backendowej serwisu, co jest niezbędne do ochrony danych użytkowników oraz zapewnienia stabilności i wiarygodności platformy.

**1.3 Charakterystyka Zawartości Rozdziałów**

W kolejnych rozdziałach przedstawiona zostanie głęboka analiza tematu crowdfundingu (Rozdział 2), szczegółowe wymagania i narzędzia wykorzystane do stworzenia platformy (Rozdział 3), a także specyfikacja zewnętrzna i wewnętrzna projektu (Rozdziały 4 i 5). Następnie, omówione zostaną procesy weryfikacji i walidacji platformy (Rozdział 6), po czym praca zostanie podsumowana, a wyniki i wnioski będą przedstawione w Rozdziale 7.

**2.** **Analiza tematu**

**2.1 Sformułowanie problemu**

W ostatnich latach obserwuje się znaczny wzrost zainteresowania crowdfundingiem oraz projektami, które wykorzystują ten model finansowania. Równocześnie zauważalny jest wzrost popularności platform internetowych wspierających crowdfunding. W trakcie analizy dostępnych serwisów crowdfundingowych zidentyfikowano lukę rynkową, którą niniejsza praca ma na celu wypełnić.

Obecne polskojęzyczne platformy crowdfundingowe często koncentrują się na zbiórkach charytatywnych lub projektach o bardziej prywatnym charakterze. Wiele z nich oferuje ograniczone możliwości prezentacji celów – zazwyczaj poprzez pojedyncze zdjęcie i tekstowy opis. W ramach tego projektu planowane jest stworzenie platformy z rozbudowanymi opcjami opisu projektu, skoncentrowanej na bardziej kreatywnych inicjatywach. Chociaż istnieją zaawansowane platformy crowdfundingowe, większość z nich dostępna jest wyłącznie w języku angielskim, co może ograniczać ich zasięg w Polsce.

Ponadto, crowdfunding wiąże się z pewnymi ryzykami, w tym z możliwością, że twórca projektu nie zrealizuje obietnic mimo zebrania funduszy. Chociaż całkowite wyeliminowanie tego ryzyka może być trudne, w projekcie przewidziano implementację rozwiązań mających na celu zminimalizowanie ryzyka oszustw i zwiększenie ochrony dla osób wspierających projekty.

**2.1 Opis istniejących rozwiązań**

W ramach analizy rynku, istniejące rozwiązania crowdfundingowe można podzielić na dwie główne grupy: międzynarodowe i polskie. Wśród polskich stron, które przeanalizowałem, znalazły się takie platformy jak polakpotrafi.pl, siepomaga.pl i wspieram.to, każda z nich charakteryzująca się swoimi zaletami i wadami.

Polakpotrafi.pl, będąca jedną z pierwszych tego typu platform w Polsce, obecnie boryka się z problemem niejednolitego interfejsu użytkownika. Siepomaga.pl, skupiająca się na akcjach charytatywnych, odbiega od celów mojej platformy, która ma koncentrować się na bardziej kreatywnych projektach.

Najbardziej zbliżona do mojej koncepcji jest platforma wspieram.to, która stosuje model kampanii "wszystko albo nic". Jednakże, ten model, mimo że bezpieczny w kontekście unikania oszustw, może stwarzać ryzyko dla projektodawców, którzy inwestują własne środki. Oznacza on że zbiórkę uznaje się za udaną dopiero od 100% celu. Jeśli zbiórka jest nieudana cały zebrany fundusz zwracany jest do inwestorów. Alternatywny model „bierzesz ile zbierzesz” na wspieram.to wiąże się ze wzrostem prowizji z 8.5% do 9.5%, co jest stosunkowo wysokim oprocentowaniem w porównaniu do innych platform. Taki system prowizji może wpływać na decyzje projektodawców oraz sukces finansowy kampanii. Chociaż ten model ma swoje ograniczenia, regularnie obserwuje się na wspieram.to udane kampanie, co świadczy o jego skuteczności w pewnych warunkach.

**3.Wymagania i narzędzia**

**3.1 Wymagania funkcjonalne**

Portal będzie miał funkcjonalność zależnie od stanu użytkownika. Rozpoznaję cztery stany, użytkownik nie zalogowany, użytkownik zalogowany z statusem USER, z statusem ADMIN i statusem SUPER.

Użytkownik nie zalogowany będzie miał jedynie dostęp do funkcjonalności publicznej. Ma dostęp do strony domowej gdzie wyświetla się kilka proponowanych projektów na podstawie jak blisko są do daty zakończenia albo te którym niewiele brakuje do osiągnięcia 100% docelowej kwoty. Może też szukać konkretnego projektu na liście projektów lub ją przefiltrować w celu przejrzenia tylko tych które mogą go zainteresować. Z strony konkretnego projektu można przejść na stronę użytkownika który dany projekt założył gdzie można znaleźć więcej informacji o tym kim jest i jego inne zbiórki. Oprócz tego mogą skorzystać z opcji która wyświetli losowy, obecnie trwający projekt oraz opcji założenia konta lub zalogowania się na już istniejącego użytkownika. W trakcie dodawania konta użytkownika sprawdzane są odpowiednie warunki takie jak czy nazwa użytkownika nie zawiera zakazanych znaków, czy email jest poprawny lub czy hasło jest odpowiednio silne.

Użytkownik typu USER ma dostęp do funkcjonalności publicznej z dodatkowymi opcjami. Przy przeglądaniu konkretnego projektu stworzonego prze innego użytkownika może on zdecydować się na wsparcie projektu wpłatą lub jeśli zdecyduje że projekt jest podejrzany może o tym poinformować administrację serwisu poprzez odpowiedni przycisk. Jeśli użytkownik jest twórcą projektu nie może go wesprzeć kwotą ani poddać pod wątpliwość jego uczciwość. Zamiast tego może go modyfikować w ograniczonym zakresie. Dodatkowo może pod projektem napisać post w którym między innymi może poinformować innych użytkowników o postępie albo inną informację związaną z projektem.

Oczywiście użytkownik typu USER może tworzyć nowe projekty poprzez odpowiedni formularz. Formularz składa się z 5 podstron. Na pierwszej podstronie użytkownik podaję nazwę projektu zdjęcie jakie chciałby zamieścić jako nagłówek zbiórki oraz może wybrać kilka z tagów/typów takich jak sztuka, muzyka czy technologia dzięki czemu łatwiej będzie trafić do grupy docelowej. Na następnej podstronie użytkownik podaję cel pieniężny jaki chciałby osiągnąć, planowaną datę końcową zbiórki oraz krótki opis. Od 3 podstronie należy wprowadzić dokładny opis celu projektu. Opis może składać się z kilku elementów którymi mogą być paragrafy tekstu lub zdjęcia. Kolejność elementów można zmieniać lub całkowicie usuwać.

Projekty są weryfikowane na dwóch etapach. Najpierw w frontedndzie gdzie sprawdzane jest czy poszczególne wartości spełniają odpowiednie wymagania. Jak na przykład czy planowana data zakończenia jest później niż dzisiejsza data albo czy nazwa projektu nie zawiera żadnych zakazanych znaków. Drugi etap odbywa się po stronie backendu gdzie sprawdzane jest czy projekt spełnia wymagana pod względem bazy danych, jeśli nie to do fronta wysyłana jest wiadomość o konkretnym będzie na podstawie której front informuje użytkownika co poszło nie tak.

Na dwóch ostatnich podstronach można dodać benefity dla osób które wesprą zbiórkę pewną kwotą i dodatkowe cele które mogą zostać dodane do efektu końcowego zbiórki jeśli całkowita zebrana kwota osiągnie odpowiednią wartość.

Po stworzeniu zbiórki musi ona być najpierw sprawdzona przez administrację serwisu czy nie łamie żadnych z zasad i czy nie wygląda zbyt podejrzanie.

Zalogowany użytkownik USER ma dostęp do strony z swoimi zbiórkami, tymi które się udały i które się nie udały, gdzie może je łatwo i szybko podejrzeć. Jeśli projekt został odrzucony przez moderację nie znajdzie się na liście publicznej projektów ale znajdzie się tutaj.

Dodatkowo każdy zalogowany użytkownik ma dostęp do strony z informacjami o swoim koncie. Gdzie może podać dodatkowe informacje o sobie lub do pewnego stopnia zmodyfikować te które już podał.

Użytkownicy typu ADMIN są administratorami serwisu. Mają dostęp do pełnej funkcjonalności publicznej. Poza tym mają dostęp do listy projektów które zostały zgłoszone prze innych użytkowników jako podejrzane. Na podstawie informacji podanych na stronie projektu lub ich braku mogą podjąć decyzje o przedwczesnym zamknięciu akcji jeśli uznają ją za dostatecznie podejrzaną. Dodatkowo jeśli znajdą użytkownika który ma złą historią mogą zablokować jego konto.

Ostatnim typem użytkownika jest typ SUPER którego jedynym celem jest dodawanie i blokowanie użytkowników typu ADMIN.

**3.2 Wymagania niefunkcjonalne**

W trakcie realizowania projektu największą wagę przykładałem do bezpieczeństwa serwisu, po stronie frontednu jak i backendu. Zaimplementowałem kilka rodzajów zabezpieczeń takich jak JWT, routr guards i własna configuracja CORS.

Dzięki JWT spora część endpointów przyjmie zapytanie tylko wtedy kiedy otrzyma także poprawny token od istniejącego użytkownika. Dodatkowo dzięki ustawieniu limitu czasu na JWT, token przypisany konkretnemu użytkownikowi ma ważność 8 godzin, po tym czasie użytkownik musi logować się ponownie do serwisu.

Router guards zapewnia że w frontendzie przekierowania miedzy stronami będą odbywać się tylko jeśli zostaną spełnione odpowiednie warunki a CORS zapewnia mogą być przyjmowane jedynie od zaufanego źródła.

Dodatkowo aby zapewnić bezpieczeństwo przesyłanych danych starałem się ograniczyć przesyłane informacje do całkowitego wymaganego minimum a takie informacje jak hasło są szyfrowane.

Aby przestrzegać zasad RODO dla użytkowników którzy nie stworzyli żadnego projektu upubliczniane są tylko i wyłącznie ich nazwy użytkowników, jeśli wpłacili na jakąś akcję organizator akcji może sprawdzić ich e-mail w celu ewentualnego kontaktu, natomiast jeśli użytkownik założy zbiórkę musi zaakceptować regulamin przez co z strony projektu będzie można się przenieść na stronę projektodawcy gdzie będzie więcej informacji o nim.

Durzą wagę przykładałem również do stworzenia intuicyjnego i łatwego w użyciu interfejsu z którego mógłby skorzystać każdy użytkownik. Ważne dla mnie było aby do każdej funkcji aplikacji można było się dostać w jak najmniejszej liczbie przejść i żeby każde przejście było świadome, jeśli użytkownik czegoś szuka to żeby od razu wiedział gdzie tego szukać. Spora część obecnie dostępnych rozwiązań strony croudfundingowej miała opcje które znajdowały się dla mnie w nie oczywistych miejscach albo opcja była nie widoczny z powodu jej umieszczenia i koloru.

Aby utrzymać dobrą jakość kodu starałem się aby był on w jak największym stopniu samogotujący się co wpłynęło negatywnie na długość nazw funkcji i klas ale pozytywnie na jego czytelność i zrozumienie. Dodatkowo starałem się aby kod był jak najprostszy do modyfikacji i diagnostyki. Dla zachowania łatwości modyfikacji kod został podzielony w taki sposób aby ułatwić wprowadzanie zmian jak na przykład oddzielenie tabeli haseł od tabeli użytkowników aby łatwiej było wprowadzić ewentualne zmiany do mechanizmu rejestracji i weryfikacji użytkownika. W celu ułatwienia diagnostyki zastosowałem Logback, logowanie informacji w springu, oraz własne błędy i mechanizm ich obsługi.

**3.3 Przypadki użycia**

**3.4 Opis narzędzi**

**3.4.1 Języki programowania**

Aplikacja stworzona jest w większości z wykorzystaniem dwóch języków programowania: Java dla strony backendowej i TypeScript dla frontendu.

W trakcie wybierania technologii do stworzenia frontendu głównie rozpatrywane były dwie opcje: JavaScript i TypeScript. JavaScript jest jednym z najbardziej popularnych narzędzi do tworzenia stron internetowych. Jest łatwo dostępny, nie wymaga żadnych dodatkowych narzędzi do uruchomienia i cechuje się dużą społecznością wsparcia, dzięki czemu istnieje wiele gotowych materiałów pomocniczych. Jego największą zaletą jest jednak prostota i szybkość tworzenia, wynikająca z dynamicznego typowania, co pozwala na szybką iterację i elastyczność w rozwoju projektów.

TypeScript, będący nadzbiorem JavaScript, został stworzony na jego podstawie i umożliwia korzystanie z czystego JavaScriptu w jego plikach. Można także używać skompilowanego kodu TypeScript w projekcie JavaScript. TypeScript rozwija ideę JavaScriptu, wprowadzając sztywne typowanie, co oznacza, że każda zmienna, parametr, czy obiekt musi mieć zdefiniowany typ. Ta cecha pozwala na lepsze zarządzanie dużymi projektami, zmniejsza ryzyko błędów wynikających z nieoczekiwanych typów danych i ułatwia współpracę w zespole programistów. Sztywne typowanie wymusza większą dyscyplinę w kodzie, co przekłada się na wyższą jakość oprogramowania i ułatwia refaktoryzację oraz testowanie.

Ostateczny wybór TypeScriptu dla frontendu został podyktowany jego zaletami w zakresie skalowalności i utrzymania dużych baz kodu. Dzięki statycznemu typowaniu, TypeScript zapewnia lepszą kontrolę nad kodem i redukuje częstość występowania subtelnych błędów, które mogą być trudne do wykrycia w JavaScript. Dodatkowo, TypeScript oferuje zaawansowane funkcje programowania obiektowego oraz kompatybilność z nowoczesnymi frameworkami front-endowymi, co czyni go idealnym wyborem dla złożonych i dynamicznych aplikacji internetowych.

Wybór Javy do stworzenie backendu był podyktowany kilkomoa kluczowymi czynnikami. Jednym z nich jest jej wydajność i skalowalność. Jest ona bowiem znana ze swojej wydajności, zwłaszcza w środowiskach wielowątkowych, co jest kluczowe dla aplikacji serwerowych wymagających obsługi wielu żądań równocześnie. Dzięki maszynie wirtualnej Java (JVM), aplikacje łatwo skalują i zoptymalizowane są pod kątem wydajności. Ta sama maszyna wirtualna JVM czyni Javę bardzo kompatybilnym i przenośnym językiem. Kod źródłowy napisany na jednej maszynie jest najpierw kompilowany do kodu bajtowego Javy, który następnie może zostać uruchomiony na dowolnej innej maszynie która ma odpowiednio zainstalowaną maszynę wirtualną Javy.

Kolejną zaletą Javy na którą zwróciłem uwagę jest jej bezpieczeństwo i niezawodność. Podobnie jak TypeScript jest to język silnie typowany co zapobiega wielu możliwym błędom w trakcie pracy programu. Dodatkowo ma mocno rozbudowany system obsługi wyjątków z możliwością definicji własnych oraz ich obsługi. Wyposażony jest również w automatyczny mechanizm zbierania śmieci dzięki czemu ryzyko jakichkolwiek przecieków pamięci jest minimalizowane.

Podsumowując, wybór TypeScriptu dla frontendu i Javy dla backendu stanowi strategiczne połączenie, które zapewnia naszej aplikacji zarówno elastyczność, jak i niezawodność. TypeScript zwiększa jakość i bezpieczeństwo kodu dzięki silnemu typowaniu, co jest kluczowe dla nowoczesnych interfejsów użytkownika. Java, z kolei, wnosi wydajność i stabilność do obsługi serwerowej, z niezawodnym zarządzaniem pamięcią i mocnym wsparciem dla wielowątkowości. Razem tworzą one solidną bazę dla rozbudowanej, bezpiecznej i łatwej w utrzymaniu aplikacji.

**3.4.2 Frameworki**

W ramach mojego projektu , kluczową rolę odgrywają nie tylko TypeScript i Java ale też dwa nowoczesne i wydajne framework stworzone dla tych języków, dla TypeScript Angular a dla Java Spring Boot. Wybór tych technologii nie jest przypadkowy, oba frameworki oferują bogaty zestaw funkcji, które znacząco przyspieszają rozwój aplikacji, a także zapewniają jej skalowalność, bezpieczeństwo i łatwość utrzymania.

Angular, wykorzystywany w warstwie frontendowej, jest frameworkiem,, oferując wszechstronne rozwiązania dla zarządzania stanem aplikacji, routingiem oraz dwustronnym wiązaniem danych. Te funkcje sprawiają, że proces tworzenia interaktywnych interfejsów użytkownika jest znacznie szybszy i bardziej efektywny. Dodatkowo, modularna architektura Angulara pozwala na tworzenie dobrze zorganizowanego i łatwego w utrzymaniu kodu, co jest nieocenione w projektach o dużym zakresie. Jednak to co uważam za największą zaletę Angulara jest możliwość pisania takich rzeczy jak instrukcje warunkowe i pętlę bezpośrednio w plikach HTML co ułatwia tworzenie dynamicznie zmieniającej się strony internetowej.

Z drugiej strony, Spring Boot, używany do budowy backendu, umożliwia szybkie tworzenie aplikacji serwerowych dzięki automatycznym konfiguracjom i prostemu wdrożeniu. Jest to idealne rozwiązanie dla szybkich iteracji i prototypowania. Spring Boot jest również doskonałym wyborem dla tworzenia bezpiecznych i łatwych w skalowaniu mikro serwisów, oferując jednocześnie wsparcie dla szerokiej gamy funkcji bezpieczeństwa.

Zarówno Angular, jak i Spring Boot są znane z ich zdolności do radzenia sobie z dużymi, złożonymi aplikacjami, oferując niezbędne narzędzia do budowy, testowania i utrzymania wydajnych i niezawodnych systemów. Wybrałem je nie tylko z powodu podanych wyżej podanych zalet ale są to też technologię z którymi mam najwięcej doświadczenia.

**3.4.3 Baza danych**

W projekcie postanowiłem wykorzystać PostgreSQL, otwarto źródłowy system zarządzania bazami relacyjnymi bazami danych (RDBMS). Cechuje się on albowiem niesamowitą rozszerzalnością poprzez dodatkowe moduły, co pozwala na implementację niestandardowych funkcji i typów danych. PostgreSQL jest także znany ze swojej wydajności, szczególnie w obsłudze dużych baz danych i skomplikowanych operacji. Mechanizmy takie jak indeksowanie (w tym indeksy GIN, **Generalized Inverted Index** i GIST, **Generalized Search Tree**), partycjonowanie tabel oraz możliwość optymalizacji zapytań, przyczyniają się do szybkiego przetwarzania danych oraz możliwości przechowywania bardziej złożonych danych w jednym polu, jak na przykład json.

**3.4.4 Środowisko i narzędzia**

W procesie tworzenia aplikacji, oprócz głównych technologii takich jak języki programowania i frameworki, kluczowe znaczenie mają również środowiska deweloperskie i narzędzia pomocnicze. W moim projekcie wykorzystuję szereg narzędzi, które nie tylko ułatwiają pisanie ale umożliwiają łatwe testowanie i zapewniają konsystencję środowiska.

**Środowiska Deweloperskie (IDE)**

* **IntelliJ IDEA Ultimate (Backend):** Dla backendu korzystam z IntelliJ IDEA Ultimate na licencji studenckiej, co jest jednym z najbardziej popularnych IDE dla Javy obecnie na rynku. Jest to kompleksowe środowisko, które oferuje zaawansowane funkcje takie jak refaktoryzacja kodu, wsparcie dla Maven i Gradle, integracja z systemami kontroli wersji oraz wsparcie dla frameworków takich jak Spring. Jego inteligentne podpowiedzi kodu i możliwości debugowania znacznie przyspieszają i ułatwiają pracę. Dodatkowo oferuje całą gamę możliwych dodatków do zainstalowania które dodają funkcjonalność która domyślnie nie była zawarta w IDE
* **Visual Studio Code:** Dla Angulara skorzystałem z VS Code ponieważ jest on szybkim i intuicyjnym edytorem tekstu którego największa zaleta leży w rozszerzeniach które można dowolnie dogrywać i konfigurować. Dzięki tym rozszerzeniom łatwo go skonfigurować pod dowolny język i technologię. Nie potrzebowałem dedykowanego IDE dla Angulara ponieważ sam Agular zawiera w sobie narzędzie ng do tworzenia i modyfikowania projektu.

**Narzędzia do testowania i zarządzania kodem**

* **Gradle:** Jako system automatyzacji budowania, Gradle jest niezbędny w moim procesie tworzenia oprogramowania. Używam go do automatyzacji zadań takich jak kompilacja kodu, zarządzanie zależnościami, pakowanie aplikacji i jej wdrażanie. Jego elastyczność i wydajność w obsłudze dużych projektów Java sprawiają, że jest to narzędzie wybrane do zarządzania cyklem budowania aplikacji. Gradle charakteryzuje się także doskonałą integracją z popularnymi IDE oraz systemami kontroli wersji, co dodatkowo ułatwia pracę nad projektem.
* **Git:** Do zarządzania wersją kodu oraz jej ewentualnego przesyłania między maszynami wykorzystałem Git, a dokładniej platformę GitHub. Pozwala to na efektywne zarządzanie kodem, śledzenie zmian i wersji, rozsyłanie między różnymi komputerami oraz przechowywanie w razie awarii lokalnej maszyny. W tym celu pomocny również był GitBash, jest to terminal który ułatwia korzystanie z komend git oraz oferuje dodatkową funkcjonalność jak na przykład wyświetlanie z jakiej gałęzi obecnie jest projekt.
* **PgAdmin:** Do zarządzania bazą danych zastosowałem narzędzie PgAdmin. Jest to zaawansowane narzędzie graficzne służące do zarządzania i projektowania baz danych PostgreSQL. Jest to jedno z najpopularniejszych narzędzi tego typu, używane zarówno przez początkujących, jak i doświadczonych administratorów baz danych. Umożliwia wizualizację struktur i relacji w bazie danych, łatwe modyfikowanie wartości w tabelach oraz wydajności bazy danych, co pomaga w identyfikacji i rozwiązywaniu problemów związanych z wydajnością i optymalizacją.
* **Postman:** Dla testowania poprawności endpitów stworzonych w backu przed przystąpieniem do ich implementacji w froncie należałoby je przetestować aby niepotrzebnie implementować coś co nie dział. Do tego celu wykorzystałem postmana, aplikację która pozwala testować wysyłanie zapytań i otzrymywanie informacji z backendu. Dodatkowo postman jest wyposażony w opcję pisania skryptów które są w stanie automatycznie sprawdzić czy odpowiedź spełnia warunki jakie zadamy.

**Docker**

Docker odgrywa kluczową rolę w zapewnieniu spójności środowiskowej i ułatwiając wdrożenia aplikacji. Aby zachować jednolite działanie aplikacji niezależnie od maszyny, na jakiej jest uruchamiana, zdecydowałem się na wykorzystanie Dockera i Docker Desktop. Oto kilka kluczowych zalet, które Docker wnosi do projektu:

* **Izolacja Środowiskowa:** Docker używa kontenerów do izolacji aplikacji, co zapewnia, że działa ona w identyczny sposób na każdej maszynie. Kontenery Docker eliminują problem „u mnie działa” poprzez zapewnienie, że wszyscy programiści pracują na tym samym środowisku, co znacznie ułatwia współpracę i testowanie.
* **Elastyczność i Skalowalność:** Docker pozwala na łatwe skalowanie aplikacji, umożliwiając uruchomienie wielu instancji kontenera bez konieczności konfigurowania dodatkowych maszyn. Jest to niezwykle przydatne w zarządzaniu obciążeniem i dostosowaniu aplikacji do zmiennego ruchu.
* **Łatwość Wdrożenia:** Docker ułatwia wdrażanie aplikacji na różnych środowiskach, od lokalnych maszyn deweloperskich po serwery produkcyjne. Dzięki zastosowaniu Dockera, proces wdrożenia jest szybki i niezawodny, co jest kluczowe w środowiskach, gdzie regularne aktualizacje i szybka reakcja na zmiany są ważne.
* **Zarządzanie Zależnościami:** Docker pozwala na definiowanie i zarządzanie zależnościami aplikacji za pomocą plików Dockerfile. Każdy kontener może zawierać wszystkie niezbędne zależności, co zapewnia, że aplikacja będzie działała poprawnie w każdym środowisku.
* **Komunikacja między kontenerami: Kiedy w projekcie takim jak ten znajduje się więcej niż jeden element korzystający z osobnego** Dockerfile można je połączyć za pomocą docker-compose, jest to plik który definiuje jakie kontenery stworzyć, jak powinny się one między sobą komunikować oraz zależności między nimi. Ponieważ mój projekt składa się z trzech kontenerów (frontend, backend i postgreSQL) docker-compose zapewnia że backend zostanie uruchomiony dopiero po pełnym uruchomieniu postgreSQL oraz to że kontenery z frontem i backiem będą się mogły między sobą komunikować dzięki docker-network.

**3.5 Metodyka pracy**

Aby dobrze pracowało się nad projektem podzieliłem jego tworzeni na kilka etapów. Na początku utworzyłem ogólny obraz aplikacji, jakie strony powinna zawierać, co na tych stronach powinno się znaleźć, jaka funkcjonalność powinna być dostępna na poszczególnej stronie i dla jakiego typu użytkownika powinna ona być dostępna. Następnie stworzyłem diagram UML reprezentujący bazę danych, jakie tabele i jakie powala powinny się w niej znaleźć oraz jakie relacje powinny zachodzić pomiędzy poszczególnymi tabelami. Na podstawie ogólnego obrazu aplikacji i diagramu UML podzieliłem funkcjonalność na odpowiednie fragmenty do implementacji i przydzieliłem każdemu z nich datę do której chciałbym dany element zaimplementować w aplikacji.

W trakcie implementacji nowego elementu w aplikacji najpierw implementowałem stronę backendu po czym ją testowałem z pokorzystaniem aplikacji Postman i PgAdmin. Po potwierdzeniu poprawności działania, lub nie działania i naprawie, przystępowałem do implementacji w Angularze. Po stronie frontendu najpierw zajmowałem się stroną kodową jak wysyłanie zapytań do backendu, konfiguracją przeniesień między stronami i tym podobne. Następnie aby łatwiej to zobrazować projektowałem rozmieszczenie poszczególnych elementów strony na podstawie czego tworzyłem odpowiednie htm i css.

Po zaimplementowaniu obu stron testowałem funkcjonalność na różnych danych wejściowych i przypadkach takich jak różny użytkownik czy inne projekty. Po przetestowaniu lokalnie uruchamiałem projekt na środowisku Docker aby zweryfikować poprawność działania aplikacji w kontenerze.

**4. Specyfikacja zewnętrzna**

**4.1 Wymagania sprzętowe i programowe**

**4.1.1 Lokalnie**

Aby uruchomić aplikację lokalnie

**4.2 Sposób instalacji**

**4.3 Sposób uruchomienia**

**4.4 Realizacja głównych przypadków użycia**

**4.4.1 Rejestracja i logowanie**

**4.4.2 Tworzenie nowego projektu jako użytkownik**

**4.4.3 Wsparcie wybranego projektu**

**4.5 Administracja serwisu**

**4.6 Bezpieczeństwo aplikacji**

**4.6.1 JWT**

**4.6.2 Mechanizm ról użytkownika**

**4.6.3 CanActivate**

**4.6.4 404**