Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut informatyki

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Inżynieria systemów informatycznych

Środowisko grywalizacji projektów studenckich

Artsiom Kashkou
Numer albumu 294529

promotor
dr inz. Mariusz Kamola

Warszawa 2022

Środowisko grywalizacji projektów studenckich

Streszczenie: Głównym celem pracy pracy jest utworzenie nowej aplikacji, która pozwoli wdrożyć element gry w proces uczenia się na podstawie monet wirtualnych. Głównymi użytkownikami tej aplikacji są studenci Politechniki Warszawskiej, którzy mogą logować się do aplikacji przez system uczelniany "USOS". Aplikacja powinna być wykorzystywana w zespole jako dodatkowe narzędzia wraz z istniejącymi. Ona ma pozwolić użytkownikom tworzyć zadanie, określać pewną nagrodę i wykonywać transakcje monet wirtualnych w przypadku, gdy autor akceptuje rozwiązanie. Kluczowymi cechami aplikacji jest wysoka szybkość działania aplikacji w przeglądarce, atrakcyjny wygląd i stos technologiczny, który zaangażuje więcej studentów we współpracę.

Słowa kluczowe: Aplikacja WEB, Grywalizacja, Kotlin, Spring, ReactJs, USOS

Gamelike environment for student projects

Abstract: The most important goal of the thesis is to create a new application with the virtual coins as an interactive element, which will improve the learning process. The main users of this application are the students of Warsaw University of Technology, who can log into the application through the university system "USOS". The application should be used in team as an additional tool along with the existing ones. It allows users to create a task, set up the reward, and receive the virtual coins when completing the task. The key features of the application are: easy and fast access to the application from the browser, modern and user-friendly design with the technological stack, which will increase the involvement of students' cooperation.

Keywords: WEB application, Gamification, Kotlin, Spring, ReactJs, USOS

Politechnika Warszawska

…….....
miejscowość i data

……..
imię i nazwisko studenta

……..
numer albumu

…….……
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta”

Spis treści

Rozdział 1

Wstęp

1.1 Motywacja i cele pracy

Zarówno na studiach uniwersyteckich, jak i w codziennej pracy zawodowej, wykonujemy zadania w ramach projektów. Tradycyjnie każdy projekt jest podzielony na zadania i to, w jaki sposób, jak szybko i skutecznie rozwiązujemy te zadania w sumie odzwierciedla poziom produktywności zespołu. W dzisiejszych czasach na rynku istnieje bardzo dużo rozwiązań dla zarządzania projektami, takie jak na przykład **Trello**[1], **Jira**[2], **Notion**[3]. Zarządzanie projektami czy zadaniami w większości polega na planowaniu, ustawieniu priorytetów i kategorii zadań, jednak jedynie zaplanowanie zadania - to za mało. Trzeba jeszcze motywować zespół do wykonania zadania. To, na ile uczestnik projektu jest zmotywowany do zrobienia zadań, jest bardzo istotne. To wpływa na szybkość i jakość wykonywania rozwiązań, czyli na produktywność. W ramach tej pracy dyplomowej będzie podjęta próba polepszania procesu wykonywania zadań poprzez wprowadzenie nagród za współpracę.

Celem jest utworzenie takiej aplikacji która będzie grała dodatkową rolę w istniejących procesach zespołu. Aplikacja ma na celu pozwolić określić nagrodę za wykonane zadanie, mianowicie ilość monet, którą można zdobyć. Monety - to nie są rzeczywiste pieniądze, to jest wirtualna jednostka. Monety wprowadzają element gry w zespole i na ich podstawie, można wymyślić różne scenariusze, w których te monety wykorzystuje się, poczynając od rankingów, czyli kto więcej monet zdobył, albo od wymiany monet na towary, które wcześniej zdefiniowaliśmy w zespole.

Kolejnym celem jest udostępnianie studentom Politechniki Warszawskiej przestrzeni, w której oni mogą współpracować i tworzyć zadania, aby uatrakcyjnić

[1] `https://trello.com`
[2] `https://jira.atlassian.com`
[3] `https://notion.so`

naukę i wprowadzić dodatkową motywację.

1.2 Związek nagrody z działaniem mózgu

W 2022 roku mamy dużo więcej informacji o działaniu naszego mózgu, niż 50, a nawet i 10 lat temu. Głównymi wrogami samokontroli podczas pracy są dopamina i serotonina, a dokładniej nieprawidłowe działanie mózgu z tymi neuroprzekaźnikami[5]. Dopamina określa poczucie oczekiwania na przyjemność — a więc motywację, ponieważ to oczekiwanie przyjemności z czegoś sprawia, że jest to coś, czego można chcieć. A u osób, które mają problemy z samokontrolą, mechanizmy dopaminy często nie działają optymalnie, na przykład powodując nadmierną chęć oderwania się od pracy i zajrzenia do sieci społecznościowych. Dopamina, to jest coś, co zmusza nas do codziennego wstawania z łóżka. Cały pomysł pracy dyplomowej toczy się wokół tego, żeby skutecznie współpracować z naszym mózgiem. I co jest bardziej istotne - to, że nie zmuszamy brutalnie naszego mózgu do tego, żeby on robił to, co jest potrzebne na studiach, czy w pracy, natomiast wykorzystujemy wiedzę, zebraną dotychczas i dążymy do tego, ludzie skupiali się na poważnych rzeczach i to dawało przyjemność.

1.3 Jak aplikacja może być pomocna w toku studiów

Istnieją również inne problemy na studiach, które można rozwiązać poprzez wprowadzenie elementów gry. Przykładowo jednym z problemów jest to, że czasami prowadzący przedmiotu ma małe zaangażowanie sprawdzanie prac studenckich. Czasami prowadzący dostaje mało odpowiedzi zwrotnej od studenta, co skutkuje spadkiem motywacji w polepszeniu procesów uczenia się.

Jednocześnie student może być mniej zainteresowany w tym żeby wgłębić się w temat i poczuć przyjemność z wykonania pracy, dlatego że nie ma jasnego celu, albo na przykład nagrody za wykonane zadanie. W innym przypadku student może popracować dobrze, ale ma trudność z otrzymaniem komentarzy od innych studentów, bo nie ma przestrzeni, gdzie można pochwalić się swoją pracą.

Z własnego doświadczenia bardzo brakowało mi przestrzeni albo narzędzia które pozwoliłoby uzyskać pomoc od innych studentów ponieważ rzadko się zdarza, że studenci chętnie pomagają innym. Aplikacja, która została utworzona w ramach tej pracy pozwala określić nagrodę za pomoc co zwiększa prawdopodobieństwo, że ktoś się odezwie i pomoże rozwiązać problem.

1.4 Opis głównych cech aplikacji

To jest przede wszystkim aplikacja webowa. *Dalej aplikacja będzie zwana jako **Coins***. Ona jest utworzona z części "frontendowej" i "backendowej", czyli

odpowiednio z części po stronie użytkownika i z części po stronie serwerowej.

Zasadniczą cechą jest to że ta aplikacja ma na celu nie zastąpić istniejące rozwiązania do zarządzania projektami i zadaniami, natomiast może być użyta jako dodatkowe, pomocnicze narzędzie, pozwalające zdobyć monety za wykonanie zadania.

Podstawowymi użytkownikami w tej aplikacji będą studenci i koordynatorzy przedmiotów Politechniki Warszawskiej. Natomiast architektura części serwerowej może być w łatwy sposób rozszerzona do tego, żeby z aplikacji korzystały inne zespoły, w których na przykład autoryzacja będzie się wykonywała przez konto Google.

Aplikacja ma na celu być prostą i zrozumiałą i zbudowaną w taki sposób żeby można było jak najszybciej nauczyć się jak z niej korzystać w zespole albo w ramach przedmiotu akademickiego.

1.5 Istniejące produkty na rynku

Jednym z pierwszych produktów, w którym spotkałem się z monetami wirtualnymi była platforma Hyperskill[4] firmy Jetbrains. Na tej platformie za każde rozwiązane zadanie można dostać monety. Otrzymane monety można wymienić na podpowiedzi za inne zadania, co radykalnie zwiększa chęć do nauki.

Kolejnym przykładem wdrożenia elementów gry jest aplikacja Forest[5]. To jest popularna aplikacja, która zwiększa produktywność w pracy. Ona pomaga ludziom pokonać uzależnienie od telefonu i zarządzać czasem w ciekawy i przyjemny sposób. Użytkownicy mogą zdobywać kredyty, nie używając swoich telefonów komórkowych i sadzić prawdziwe drzewa na całym świecie za pomocą kredytów. Dzięki lasowi użytkownicy mogą mieć wspaniałe doświadczenie, aby spędzać mniej czasu na telefonach komórkowych, skupiać się na tym, co jest ważniejsze w ich życiu.

Habitica[6] - to jest darmowa aplikacja dla rozwoju nawyków i produktywności, która traktuje prawdziwe życie jak grę. Habitica może pomóc osiągnąć cele, aby stać się zdrowym i szczęśliwym.

Pointgram[7] - to jest aplikacja, która z kolei skupia się na motywacji, tworząc system punktów, przyznając odznaki, wyświetlając konkursy, przydzielając zadania i otwierając sklep z nagrodami.

Xoxoday[8] jest aplikacją z szerokim zakresem funkcjonalności. Ona jest przeznaczona dla używania w dużych zespołach, jest mocno konfigurowalna i ma możliwość uruchomienia na własnych serwerach w niezależnych organizacjach.

[4] `https://hyperskill.org` - platforma edukacyjna
[5] `https://www.forestapp.cc` - aplikacja dla zarządzania czasem pracy
[6] `https://habitica.com` - aplikacja dla spełniania celów z elementami gry
[7] `https://www.pointagram.com`
[8] `https://solutions.xoxoday`

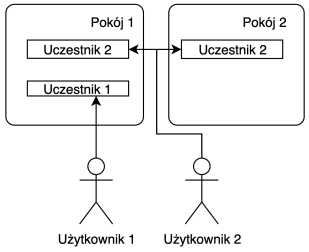
Natomiast produkt, który powstaje w ramach tej pracy, będzie przede wszystkim przeznaczony dla uczelni i powiązany z systemem USOS. Ten system, z kolei, pozwoli uwierzytelniać użytkowników, tworzyć unikatowe scenariusze, których nie dadzą wyżej wymienione rozwiązania.

Rozdział 2

Realizacja pracy

Realizacja pracy rozpoczyna się od określenia dziedziny problemu i wymagań, które muszą być spełnione. Dalej będą wybrane niezbędne technologie i na ich podstawie powstaną rozwiązania, które będą spełniali wymagania.

2.1 Słownik



Rysunek 2.1: Słownik

- Użytkownik - reprezentuje jedną osobę, która jest zarejestrowana w aplikacji **Coins**
- Pokój - przestrzeń w której tworzą się zadania przez uczestników pokoju
- Uczestnik - reprezentuje użytkownika, który został dodany do pokoju
- Portfel - reprezentuje portfel uczestnika

W systemie **Coins** każdy użytkownik może być dodany do jednego lub więcej pokojów. Gdy użytkownik dodaje się do pokoju, on staje się jego uczestnikiem. W ramach jednego pokoju uczestnik ma jeden unikatowy portfel. Uczestnik może utworzyć zero lub więcej zadań, które będą widoczne jedynie w pokoju w którym te zadania zostały utworzone. Uczestnik może określić nagrodę, czyli

ilość monet za wykonane zadanie i wtedy wykonawca może otrzymać nagrodę na swój portfel.

2.2 Wymagania konieczne

- Aplikacja ma umożliwić definiowanie zadań w różnych pokojach do których użytkownik ma dostęp.
- Powinien poprawnie działać scenariusz, w którym autor tworzy zadanie, następnie uczestnik rozwiązuje go, zlecający akceptuje rozwiązanie i w wyniku nagroda w postaci monet przydziela się użytkownikowi, który rozwiązywał zadanie.
- Ma być wdrożone uwierzytelnianie użytkowników przez uniwersytecki system USOS.
- Muszą istnieć dwie role użytkowników: Użytkownik zwykły i administrator.
- Administratorzy powinni mieć dostęp do panelu administratora, w którym da się utworzyć nowe pokoje i definiować początkową zawartość portfela użytkownika.
- Ma być wdrożona funkcjonalność generacji raportu dla podanego pokoju, w którym będzie widoczne ile monet zdobył każdy użytkownik.

2.3 Wymagania dodatkowe

- W panelu administracyjnym pokój może być utworzony na podstawie przedmiotu z platformy **USOS**
- W panelu administracyjnym ma istnieć funkcjonalność, odpowiadająca za automatyczne dodawanie wszystkich uczestników przedmiotu do nowego pokoju.
- Podczas tworzenia zadania w aplikacji, użytkownicy mają możliwość powiązania treści nowego zadania z zadaniem, które zostało utworzone w serwisie Github[1].

2.4 Technologie i narzędzia

Każda podjęta decyzja na temat technologii, narzędzi i języków programowania opiera się na trzech kluczowych pytaniach (Źródło[2]).

[1] `https://github.com` - Github

[2] `https://youtu.be/Z0yZgqqy5Lc` - Making Good Tech Decisions - MY FIRST CONFERENCE TALK

Pierwsze pytanie - to na ile kosztowne jest wdrożenie jakiejkolwiek technologii, szczególnie jaki jest jej koszt utrzymywania w ramach zespołu. Na przykład na ile ciężko nauczyć nowych członków zespołu języku programowania i kosztowne to będzie dla zespołu w przypadku, gdy trzeba będzie zrezygnować z podjętych decyzji.

Drugim, nie mniej istotnym czynnikiem jest oszacowanie na ile podjęta decyzja jest ryzykowna. Żeby oszacować ryzyko trzeba przynajmniej zwrócić uwagę na to, czy technologia jest popularna i niezawodna, czyli jakie jest wsparcie i kto pracuje nad technologią. Trzeba też wziąć pod uwagę na ile jest ona zrozumiała szczególnie dla ludzi, którzy nie spotykali się z nią wcześniej.

Ostatnią rzeczą, na którą warto zwrócić uwagę, to jest oszacowanie wartości, którą daje wybór tej czy innej technologii. To jednocześnie może być poziom satysfakcji z używania tej technologii, ilość czasu jaki można oszczędzić w porównaniu do podobnych produktów. Oraz na ile jest prawdopodobne, że użytkownik będzie miał więcej satysfakcji, gdy realizacja projektu będzie się opierała na konkretnej technologii.

2.4.1 Języki programowania

Część Serwerowa

Część serwerowa została napisana w jednym z języków, generujących kod interpretowany przez maszynę wirtualną **"Java Virtual Machine"**. Bardzo popularnym wyborem, gdy chodzi o stronę backend, jest język Java. I w tym języku jest napisana olbrzymia ilość oprogramowania. Zgodnie z raportem (Źródło[3]), w projektach opartych na Java Virtual Machine, język Java wykorzystuje się w 90% przypadkach, natomiast język Kotlin jest na drugim miejscu na liście popularności i on wykorzystuje się w 17.7% przypadkach. W porównaniu z oficjalnym raportem twórców języka Kotlin (Źródło[4]), Kotlin wykorzystano w 5 procent projektów w roku 2020. Niemniej jednak wybór padł na język Kotlin.

Kotlin bardzo mocno się rozwija i jest to jeden z kluczowych produktów firmy **Jetbrains**. Jetbrains jest znana jako najbardziej popularna firma technologiczna na rynku. Każdy programista słyszał o narzędziach **Intellij**, pomagających pisać projekty w wygodny sposób i mieć sporo inteligentnych wtyczek, które oszczędzają czas pracy. Więc ta firma daje bardzo mocne gwarancje, że jej produkty nagle nie zginą z rynku.

Część frontendowa

Po stronie frontendowej, w większości przypadków prawie nie mamy lepszego wyboru niż język **JavaScript**. Istnieją różne podejścia takie jak na przykład robienie zarówno backendu, jak i frontendu w jednym języku języku. Albo używanie języków funkcyjnych, takich jak na przykład **Elm**, **Rescript**, albo **Clo-**

[3] `https://res.cloudinary.com/snyk/image/upload/v1623860216/reports/jvm-ecosystem-report-2021.pdf` - jvm-ecosystem-report-2021

[4] `https://www.jetbrains.com/lp/Kotlin-census-2020/` - Kotlin-census-2020

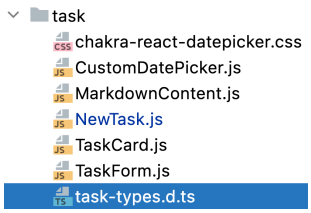
jureScript. Jednak każdy inny wybór w porównaniu do JavaScript drastycznie przegrywa na podstawie wcześniej wymienionych kluczowych czynników przy wyborze technologii. Czyli ciężko będzie znaleźć nowych ludzi którzy będą w stanie dołączyć do projektu, jeśli będzie wybrany język, który jest dużo mniej popularny niż JavaScript. Albo budowanie aplikacji WEB może stać się czasochłonne, jeśli ona będzie napisana w języku Kotlin. Bo to podejście nie jest na razie popularne i stabilne. Jest mniej narzędzi i materiałów na temat tego w jaki sposób zbudować aplikacje przeglądarkowe wyłącznie w języku Kotlin.

Nie można jednak zapomnieć o innym trendzie 2022 roku, czyli o języku **TypeScript** (Źródło[5]). Język TypeScript wywodzi się z języku JavaScript i wprowadza typowanie na etapie kompilacji. Jednak istnieje kilka powodów, dla których ten język nie stał głównym językiem w projekcie. Przede wszystkim, twórcy TypeScript oficjalnie mówią na swojej stronie z dokumentacją, że ten język nie daje stuprocentowej gwarancji że wszystkie błędy związane z typowaniem, rozwiążą się samoczynnie. To dlatego, że TypeScript nadal może działać w połączeniu z JavaScript, natomiast JavaScript nie jest językiem statycznym, to jest język dynamiczny i on nadal interpretuje się przez przeglądarkę i TypeScript kompiluje się w JavaScript.

Niemniej jednak, jeden dosyć przydatny element z języka TypeScript został dodany do projektu, mianowicie tak zwane pliki **.d.ts**. To są pliki, w których deklaruje się typy i interfejsy. Programiści piszący w języku JavaScript zwykle opisują typy w komentarzach dokumentacji **JsDoc**. I na podstawie tych komentarzy można korzystać z inteligentnych sugestii i analizy kodu we współczesnych edytorach kodu. Zaletą plików **.d.ts** jest to, że typy zadeklarowane w tych plikach mogą być wykorzystywane w dokumentacji JsDoc. Na przykładzie poniżej zostało pokazane jak z tego można korzystać.

Niech zadaniem będzie utworzenie typu "Zadanie" i funkcji pomocniczej, która będzie zwracała termin ostateczny wykonywania zadania.

Najpierw utworzymy interfejs **TNewTask** w pliku *task/task-types.d.ts*



Rysunek 2.2: Katalog

Następnie zadeklarujemy funkcję **extractDedaline()** i skoro opisaliśmy typ w dokumentacji, obserwujemy, że środowisko **Intellij Idea** podpowiada jakie są pola w zmiennej **task**.

[5] `https://youtu.be/CGnlBU3K_eM` - The third age of JavaScript: Three years in - Swyx

```
interface TNewTask {
  title: string
  content: string
  deadline: Date
  budget: number
}
```

Rysunek 2.3: Interfejs

Próbując zwrócić zmienną **deadline**, środowisko podpowiada, że typy nie pasują do siebie z tego podowu, że zmienna deadline ma typ **Date**, natomiast funkcja **extractDeadline** celowo zwraca typ **number**.

Na koniec, po zmianie typu, zwracającego przez funkcję dostajemy w wyniku poprawnie zwrócony typ i zero błędów w podpowiedziach

```
/**
 * Extract deadline from the task
 * @param {TNewTask} task
 * @return {number}
 */
function extractDeadline(task: TNewTask) {

}
```

Rysunek 2.4: Deklaracja

2.4.2 Wybór technologii

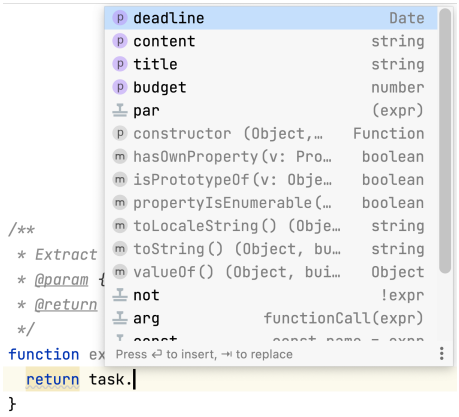
Technologie serwerowe

W dalszej części pracy często będą wspomniane pojęcia biblioteki i frameworku (albo architektura). W ramach tej pracy różnica pomiędzy tymi dwoma pojęciami jest szczegółowo opisana w tym źródle[6]. Na potrzeby zrozumienia treści pracy dyplomowej, wystarczy przyjąć, że

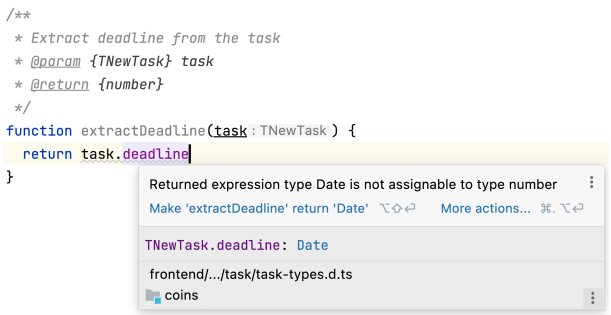
- Biblioteki zapewniają programistom predefiniowane funkcje i klasy, aby ułatwić ich pracę i przyspieszyć proces rozwoju
- Framework, z kolei, jest fundamentem, na podstawie którego programiści budują aplikacje dla konkretnych platform. Framework narzuca pewne reguły, styl projektowania aplikacji

Po pierwsze, cały serwer został oparty na najbardziej popularnym i de-facto standardowym frameworku ze świata języka Javy, na frameworku **Spring**. Skoro

[6] `https://www.interviewbit.com/blog/framework-vs-library` - Framework vs library



Rysunek 2.5: Sugestie



Rysunek 2.6: Niepoprawne

```
/**
 * Extract deadline from the task
 * @param {TNewTask} task
 * @return {Date}
 */
function extractDeadline(task : TNewTask ) {
  return task.deadline
}
```

Rysunek 2.7: Poprawne

wybrany język Kotlin jest w 100% kompatybilny z językiem Java, a jednocześnie Spring jest w pełni obsługiwany przez Kotlin, wtedy taki wybór architektury bardzo dobrze odpowiada specyfice projektu. Spring rozwija się od 2003 roku i w nim zostały zaimplementowane tysiące komponentów, pozwalających rozwiązać większość problemów, z którymi każdy spotyka się przynajmniej na starcie projektu. O tym jakie są możliwości tego frameworku, można zapoznać się na jego oficjalnej stronie, dalej zostaną wymienione bazowe elementy, realizowane przez Spring:

- Uruchomienie **Serwera Apache Tomcat**[7]
- Zarządzanie kontenerem komponentów i cyklem ich życia
- Połączenie z bazą danych
- Konfiguracja kontrolerów i adresów URL
- Konfiguracja uwierzytelniania

Również były podjęte próby budowania projektu na podstawie frameworku Ktor[8]. To jest nowoczesne rozwiązanie, jedynie działające z językiem Kotlin, jego pierwsza wersja powstała w 2018 roku. Ktor, jak każda inna technologia, ma swoje wady i zalety. Jednym z najbardziej istotnych powodów, dlaczego ten framework nie zastąpił Spring - to jest poziom ryzyka, który wiąże się wyborem Ktor. Ryzyko rośnie dlatego, że nie mając doświadczenia z tą technologią, bardzo ciężko oszacować w którym momencie zabraknie funkcjonalności, która istnieje w Spring i nie istnieje w Ktor. Wybór Ktor, może być uzasadniony, gdy w zespole mogą być przydzielone dodatkowe środki i czas na to, żeby osiągnąć funkcjonalność, na podstawie której będzie napisana logika biznesowa. Ilość dostępnych materiałów też gra istotną rolę i podobnie do innych technologii z których zrezygnowałem, ciężko byłoby zagwarantować łatwość dalszego rozwoju projektu.

Ktor ma swoje zalety, mocno opiera się na konstrukcjach językowych Kotlin, których nie ma w języku Java, co pozwala w dosyć wygodny i deklaratywny sposób konfigurować serwer i budować projekt. Inną zaletą może być to, że we frameworku Ktor jest wzięty pod uwagę inny trend, mianowicie asynchroniczny paradygmat programowania. W języku Kotlin jest możliwe pisanie kodu asynchronicznego na podstawie specjalnych funkcji, które mogą być wstrzymywane w pewnych punktach (suspension points) i potem kontynuowane. Ale z tym wzrasta poziom złożoności dla programistów, którzy nie mieli z tym do czynienia wcześniej.

Bazy danych

Kwestia zarządzania bazami danych gra również kluczową rolę w aplikacjach backend-frontend. W pierwszym kroku trzeba się zastanowić jaki rodzaj bazy będzie wspierany w projekcie, czy to będzie baza relacyjna albo jakikolwiek inny

[7] Apache Tomcat - to jest standardowy serwer, który uruchamia się przez Spring Framework
[8] `https://ktor.io` - Biblioteka Ktor

rodzaj bazy nierelacyjnej. Dziedzina projektu nie jest specyficzna i nie wymaga projektowania złożonej funkcjonalności, opartej na modelach na przykład w postaci grafu. Baza relacyjna bardzo dobrze pasuje do problemu również z uwagi na niskie wymagnia dotyczące jej wydajności. Dodatkowym walorem jest to, że zarówno framework Spring, jak i inne biblioteki, dostępne dla języków programowania z paradygmatem obiektowym, pozwalają generować bazę i wykonywać polecenia całkowicie z poziomu kodu serwerowego, na podstawie instancji klas i specyficznych konstrukcji języka. Czyli klasy z części serwerowej mogą przekładać się na tabele baz danych. To podejście jest znane jako Object-relational Mapping (ORM). W większości przypadków, projekty, które są oparte na frameworku Spring korzystają z biblioteki Hibernate. Ona automatycznie generuje tabele bazy danych na podstawie klas w języku JVM, czyli na przykład Kotlin.

Wybór bazy danych zwęził się do dwóch kandydatów: **PostgreSQL**, **MySQL**. To dlatego że to są bazy relacyjne z podstawowym językiem SQL i one dobrze pasują do dziedziny aplikacji. Dla danego projektu różnica pomiędzy tymi bazami jest ledwie zauważalna. PostgreSQL i MySQL mają bardzo podobne dialekty SQL i możliwości funkcjonalne. Te bazy można porównywać w szerokim zakresie, w projektach, gdzie jednocześnie tysiące zapytań wykonują się na raz. W przypadku gdy projekt ma taką specyfikę, że czyta się dużo więcej rekordów niż zapisuje się, wtedy MySQL działa w większości przypadków szybciej (Źródło[9]). PostgreSQL również jest bardzo zoptymalizowany, obsługuje indeksowanie dla kolumn typu JSON i na ogół wspiera więcej typów niż MySQL. PostgreSQL został wybrany jako jedyna potrzebna baza danych w projekcie.

W kolejnym kroku należy wybrać sposób komunikacji z bazą danych z poziomu kodu serwerowego.Warto odnotować, że połączenie Spring i Hibernate jest bardzo wygodne a zaletą jest prostota tworzenia schematu bazy i wygoda wykonywania poleceń. Żeby uzyskać taką prostotę, Hibernate bierze na siebie odpowiedzialność za wykonywanie poleceń, lecz za tym idzie bardzo mocna wada. Można prześledzić styl kodowania i sposób w jaki wykonują się polecenia. On jest mocno powiązany z klasami, reprezentującymi tabele w bazie danych. Te klasy nie są rodzaju "Immutable classes" i ich instancje mogą zmieniać swój stan w trakcie działania aplikacji. Instancje tych klas niejawnie są zarządzane przez mechanizmy Spring i Hibernate, co powoduje to, że programista powinien za każdym razem mieć pewność, że kod, który się pisze jest zgodny z intencjami. Brak testów sprawdzających poprawność wykonywania poleceń powoduje zwiększenie ryzyka, że w pewnym momencie zespół nie będzie w stanie wykryć dlaczego aplikacja działa niepoprawnie i w jakim stanie ona się znajduje.

Z tych powodów dużo lepiej prezentuje się biblioteka **JOOQ**. Wdraża ona inny sposób współpracy z bazą danych. Żeby wykonywać polecenia, biblioteka zawiera zestawy operacji, które jawnie odzwierciedlają język SQL i do tego zapewnia że każde polecenie będzie poprawnie typowane. I to jest podstawowe wymagania projektu - mieć pewność, że polecenie które będzie napisane z poziomu języka

[9] `https://blog.devart.com/postgresql-vs-mysql.html` - Porównanie Postgresql a mysql od firmy Devart

Kotlin będzie odwzorowane na polecenie SQL bazy danych. Programista powinien jawnie wskazać jaki rekord on chce odczytać albo jakie dane konkretnego typu trzeba wstawić do tabeli bazy danych. Eksperymenty z tą biblioteką zakończyły się pomyślnie, a JOOQ został wybrany jako podstawowa technologia która łączy kod serwerowy z bazą danych.

Technologie dla aplikacji przeglądarkowej

Dziedzina frontendu i JavaScriptu w dzisiejszych czasach jest mocno powiązana z architekturami **ReactJS**, **VueJs**, **AngularJs**. To jest trójka najbardziej kluczowych porduktów na rynku i w zależności od wyboru technologii projekty będą mocno się różniły, zaczynając od struktury projektu i kończąc na konwencjach, nazewniczych w kodzie. ReactJS został wybrany jako główna technologia frontendowa, dlatego że po prostu zdobyłem dużo więcej doświadczenia z ReactJS niż z innymi rozwiązaniami. Każde z nich ma prawie wszystkie niezbędne dodatkowe biblioteki zarówno z otwartą bazą kodu (znane jako Open Source) jak i zamkniętą.

Do napisania dobrej aplikacji działającej w przeglądarce nie wystarczyło dodać jedynie bibliotekę ReactJS. Dlatego że istnieją przynajmniej trzy dziedziny, w których trzeba dołączyć dodatkowe technologie. Mianowicie, żeby tworzyć formularze w standardowy i wygodny sposób, w ramach projektu używa się biblioteki "React hook form". Jako technologia pozwalająca tworzyć zapytania do serwera i przechowywać odpowiedzi (znane jako caching) zastosowana jest biblioteka "React Query". Ostatnim niezbędnym elementem jest bbilioteka "Chakra UI", która pomaga tworzyć nowoczesny interfejs.

2.4.3 Wybór narzędzi

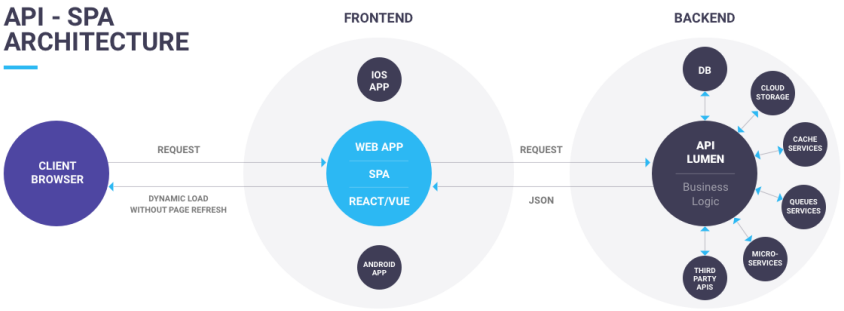
Środowisko **Intellij Idea Ultimate** nie pozostawia wyboru, gdyż jest to jedyna niezbędna aplikacja dla tego projektu w której zarządza się kodem i strukturą projektu. Wersja Ultimate pozwala zarządzać wieloma bazami danych naraz i korzystać z wielu innych funkcjonalności, takich jak uruchomienie skryptów SQL i modyfikacja zawartości i struktury tabel. Istotne jest to, że to narzędzie należy do firmy Jetbrains, czyli tej firmy, która rozwija język Kotlin. Niestety nie istnieją inne narzędzia, pozwalające uzyskać nawet podobną ilość funkcjonalności jaką daje Intellij Idea. Do tego twórcy tego środowiska wzięli pod uwagę wszystkie najbardziej popularne języki i biblioteki na podstawie których pisze się oprogramowanie i dlatego tutaj też "Idea" jest bezkonkurencyjna. Algorytmy zaimplementowane w tej aplikacji mogą skanować strukturę projektu i specyficzne komponenty, które należą do wybranych bibliotek. To daje dodatkowe sprawdzanie błędów, sprawdzanie czy struktura komponentów jest poprawna, oraz automatyczną generację kodu, który często się powtarza.

2.5 Architektura aplikacji

Podczas projektowania architektury zostały uwzględnione najbardziej popularne i skuteczne podejścia wśród znanych architektur backend-frontend. Przykładem takich reguł jest metodologia "Twelve-Factor"[4]. Podaje ona dwanaście reguł dla budowania aplikacji webowych, które:

- używają deklaratywnych formatów do automatyzacji konfiguracji, aby zminimalizować czas i koszty dla nowych programistów dołączających do projektu;
- korzystają z czystego kontraktu z bazowym systemem operacyjnym, oferując maksymalną przenośność między środowiskami wykonawczymi;
- nadają się do wdrożenia na nowoczesnych platformach chmurowych, eliminując potrzebę administrowania serwerami i systemami;
- minimalizują dysproporcje między rozwojem a produkcją, umożliwiając ciągłe wdrażanie w celu uzyskania maksymalnej sprawności;
- umożliwiają skalowanie bez znaczących zmian w narzędziach, architekturze lub praktykach programistycznych.

Również zostało uwzględnione to, że wcześniej wybrany stos technologiczny częściowo narzuca własne reguły projektowania i pewien styl kodowania aplikacji.



Rysunek 2.8: Architektura (źródło: PHP API-SPA Architecture for The ultimate User Experience)

Na rysunku została pokazana architektura aplikacji Coins, ona jest typu Single Page Application (SPA) i komunikuje się z serwerem. Aplikacja działa w następujący sposób: przeglądarka wyświetla stronę frontendową, która jest zbudowana na podstawie biblioteki ReactJs. Żeby wyciągnąć dane z serwera, tworzy się zapytanie i po dostaniu odpowiedzi, zawartość strony odświeża się dynamicznie, czyli bez ponownego ładowania całej strony. Dla przykładu, na rysunku jest pokazane, że część frontendowa może również być napisana na podstawie biblioteki

"Vue" albo w ogóle zbudowana w postaci aplikacji mobilnej. Natomiast część serwerowa również może komunikować się z innymi serwerami i chmurami.

2.5.1 Część serwerowa

Skoro po stronie serwerowej zostały wybrane technologie: język Kotlin i framework Spring, to struktura projektu, jak i organizacja komponentów również jest zgodna z przykładami udostępnionymi na stronie oficjalnej Spring (Źródło[10]). W tym frameworku zostało przyjęte, że komponenty budują się zgodnie z projektowaniem w stylu: Domain Driven Design (DDD). Komponenty mają nazewnictwo zgodne z DDD i z ustalonymi zasadami, dzięki czemu można w dosyć wygodny i stabilny sposób korzystać z istniejących rozwiązań, utworzonych przez twórców frameworku Spring.

Jednak opanowanie reguł DDD jest często kłopotliwe, pojawiło się wiele książek, takich jak Domain-Driven Design: Tackling Complexity in the Heart of Software[1], albo na przykład Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy[3]. Natomiast z mojego doświadczenia wystarcza zredukować ten rodzaj projektowania do zasad, które w większości świetnie opisują się w architekturze Hexagonalnej[2]. Wszystkie powyższe architektury łączą następujące zasady:

- grupowanie komponentów w warstwy
- projektowanie domeny biznesowej, czyli encji, które występują w problemach projektowych
- projektowanie stabilnych abstrakcji i na ich podstawie tworzenie konkretnych implementacji

Dlatego aplikacja została podzielona na warstwy, gdzie podstawowymi warstwami są po kolei: warstwa prezentacji, warstwa serwisowa i warstwa bazy danych. Dalej, zostaną wymienione istotne reguły, na podstawie których wykonano warstwy aplikacji.

Warstwa prezentacji (Presentation layer)

Warstwa prezentacji jest oparta na kontrolerach, które udostępniają interfejs REST API. Stąd klientem może być jakakolwiek aplikacja, na przykład zarówno aplikacja mobilna jak i aplikacja, działająca w przeglądarce. Kontrolery starają się dostarczać klientom stabilne API.

Skoro warstwa prezentacji jest połączona z warstwą serwisową wtedy niezbędne jest zagwarantowanie, że tylko potrzebne dla użytkownika dane będą udostępnione. W szczególności najbardziej istotne jest to, żeby ukryć prywatne dane, wychodzące z warstwy serwisowej. Na przykład model użytkownika, który był przygotowany przez warstwę serwisową dla warstwy prezentacji może zawierać hasła i żeby ukryć te hasła, korzystamy z "Data transfer objects" (DTO). Czyli

[10] `https://spring.io/guides/gs/spring-boot/` - Spring boot guide

to są obiekty, które w ramach założonej architektury, mają jedną rolę: transportować dane pomiędzy warstwami. Są kilka powodów dlaczego DTO są istotne:

- sprawdzają poprawność danych wejściowych, na przykład za pomocą adnotacji Bean Validation dla klas DTO
- egzekwują ścisłe granice API
- zapewniają stabilne API dla klientów, które nie zepsuje się podczas refaktoryzacji wewnętrznych modeli aplikacji
- mogą zapewniać optymalizację dla warstwy prezentacji, zwracając DTO zawierające tylko te atrybuty, które są absolutnie wymagane

W przypadku gdy serwis potrzebuje przyjąć dany model warstwy prezentacji, konwencja zakłada, że te obiekty mają przyrostek "Options", na przykład UserOptions.

Standardowe operacje dla zasobów, którymi można zarządzać z REST API:

- GET wszystkie zasoby (z sortowaniem i stronicowaniem, jeśli jest potrzebne)
- GET zasób na podstawie identyfikatora
- POST zasób, żeby utwrorzyć nowy
- PUT zasób, żeby zastąpić wszystkie istniejące pola
- PATCH zasób, żeby odświeżyć tylko nowe pola
- DELETE zasób

Warstwa usług (Service layer)

Ona współpracuje z klientami biznesowymi (Business entities), deklaruje granice systemu, organizuje transakcje. Usługi reprezentują bezstanowe obiekty mające zestaw procedur, gdzie każda procedura wykonuje odrębny kawałek logiki. Ta warstwa przekazuje dane pomiędzy warstwą bazy danych a prezentacji.

Warstwa bazy danych

Istotną cechą warstwy bazy danych jest to, że ona zawiera tylko logikę zapytań do bazy danych. W projekcie została opracowania tak, żeby w przypadku, gdy będzie podjęta decyzja zrezygnować z biblioteki JOOQ albo zastąpić bazę PostgreSQL inną bazą, wszystkie pozostałe warstwy, mianowicie warstwy serwisowe i prezentacji pozostaną niezmienione. Jedynie trzeba będzie utworzyć nowe implementacje istniejących interfejsów warstwy bazy danych.

Jedyny obiekt, który jest potrzebny żeby rozpocząć wykonywanie zapytań do bazy, to obiekt z biblioteki JOOQ "DSLContext". W zasadzie wystarczy odwoływać się do tego obiektu i to pozwoli pobierać, i przechowywać dane w bazie. Jednak skutkiem tego podejścia jest to, że w przypadku, gdy trzeba będzie zmienić bibliotekę JOOQ na inną, wtedy większość warstwy serwisowej może zostać

przepisana. Wynika to stąd, że nie istniało interfejsów, które zawierałyby zestawy operacji, powiązanych z bazą danych. Dlatego bardzo zaleca się wprowadzać takie interfejsy, których implementacje będą zwane jako "Data Access Objects" (DAO).

Obiekty DAO dalej będą traktowane po prostu jako obiekty, które mają metody, pozwalające wykonywać fragment logiki, odwołujący się do bazy danych. Na przykład w obiekcie, który jest powiązany z zadaniami "TaskDao", takie metody mogą być nazwane jako "fetchTaskById()", żeby pobierać zadania albo na przykład "save(task)", żeby je przechowywać je. We frameworku Spring przyjęto nazywać takie obiekty jako "Repository", natomiast w bibliotece JOOQ jest przyjęte nazewnictwo DAO zamiast Repository i to będzie traktowane jako główne nazewnictwo w projekcie.

2.5.2 Część frontendowa

W obszarze frontendu istnieje mnóstwo zagadnień i problemów które mogą być rozwiązane za pomocą dostępnych pakietów (NPM packages), które można dołączyć do projektu za pomocą menedżera pakietów "NPM". Mowa o architekturze frontendowej zwykle podchodzi się do trzech kluczowych zagadnień. Mianowicie: w jaki sposób wyświetlać komponenty, jak zarządzać stanem tych komponentów, oraz w jaki sposób wykonywać zapytania do serwera.

Wybrana biblioteka ReactJS, definiuje pewien styl kodowania i pozwala wyświetlać komponenty, które definiują się w plikach ".jsx". To są pliki JavaScript, mające dodatkową cechę: mogą zawierać kod HTML, a jednocześnie kod JavaScript. To rozwiązuje problem wyświetlana (renderingu) komponentów, bo ReactJS bierze na siebie odpowiedzialność za generację dokumentów HTML.

Kolejną kwestią jest zarządzanie stanem komponentów i znów biblioteka ReactJS proponuje dwa podejścia do zarządzania stanem. W 2018 roku zostało przedstawione nowe, dzisiaj w większości przypadków standardowe, podejście, oparte na technologii "React Hooks". Zarówno do 2018 roku, jak i dziś można tworzyć komponenty na podstawie klas JavaScript. Te klasy jednocześnie zawierały zmienne dotyczące stanu obiektu a metodę "render", która zwracała kod, wcześniej wspomnianego formatu JSX, czyli pewien rodzaj HTML. Ale w tym projekcie wykorzystałem nowe podejście oparte na "Hookach". W tym przypadku całe komponenty są zawarte po prostu w funkcjach JavaScript, a nie w klasach, i wartość, zwrócona przez tą funkcję, odzwierciedla metodę render, która istniała w poprzednim podejściu. Taka forma komponentów wiąże się z tym, że twórcy biblioteki ReactJS podjęli decyzję, że jednak techniki, istniejące w programowaniu funkcyjnym, mogą być również wykorzystane w jeżyku JavaScript i da się utworzyć taką samą funkcjonalność, opartą na klasach, która w większości przypadków tworzy mniej złożone rozwiązania.

Hooki, to jest nic innego jak funkcje JavaScript. Kluczowe jest to, że hooki można wykorzystywać w wielu komponentach, bo znów to są po prostu funkcje. Trzeba mieć inny styl myślenia, żeby przestać projektować w ramach klas

JavaScript i zacząć projektować zgodnie z filozofią, opartą na hookach.

Jak już było wspomniane trzecia biblioteka która wzbogaca funkcjonalność ReactJS - to jest "React Query". Ona bierze na siebie odpowiedzialność za wprowadzenie stanu podczas wykonywania zapytań do serwera. Taka biblioteka została dołączona do projektu dlatego, że każde zapytanie do serwera musi być zrobione w taki sposób, żeby umożliwić śledzenie czy zapytanie wykonało się poprawnie, czy serwer zwrócił jakąkolwiek błędną odpowiedź, czy klient już czeka długo na odpowiedź i tak dalej. Czyli ta biblioteka automatycznie tworzy niezbędne zmienne stanowe i w ten sposób nie trzeba ręcznie pisać powtarzający się kod.

Ale to nie jest główny powód dlaczego wrato zwrócić uwagę na "React Query". Gdy wykonujemy zapytania, czasami istnieje potrzeba przechowywania danych (caching) z odpowiedzi po stronie klienta. Caching - to jest niezbędna rzecz, żeby użytkownik czuł płynność i wygodę w używaniu aplikacji. Gdy aplikacja będzie pobierała takie same dane z serwera, które nie zmieniają się w zależności od podanych argumentów, wtedy klient będzie musiał czekać na załadowanie tych danych za każdym razem. Przechowując te dane nie musimy ponownie robić zapytania do serwera i mamy możliwość tworzenia wielu komponentów, mających dostęp do tych danych za pomocą mechanizmu "Caching".

2.5.3 Rozszerzenie Microsoft Teams jako próba robienia aplikacji klienta

Skoro aplikacja została wcześniej podzielona na część kliencką i serwerową, a serwer udostępnia interfejs REST API, który nie zależy od konkretnej implementacji klienta, na początku rozwoju projektu była podjęta próba wytworzenia jej w postaci rozszerzenia platformy Microsoft Teams. Głównym celem tego kierunku działań było sprawdzenie czy takie rozszerzenie może zastąpić aplikację w odrębnym wydaniu przeglądarkowym. Istniało kilka powodów powstania takich eksperymentów integrujących z MS Teams. Przede wszystkim jednym z wymagań projektu było udostępnianie użytkownikom takiego interfejsu, albo inaczej mówiąc, takiej aplikacji, która będzie obejmowała jak najmniej kroków do tego, żeby zacząć korzystać z zaimplementowanej funkcjonalności.

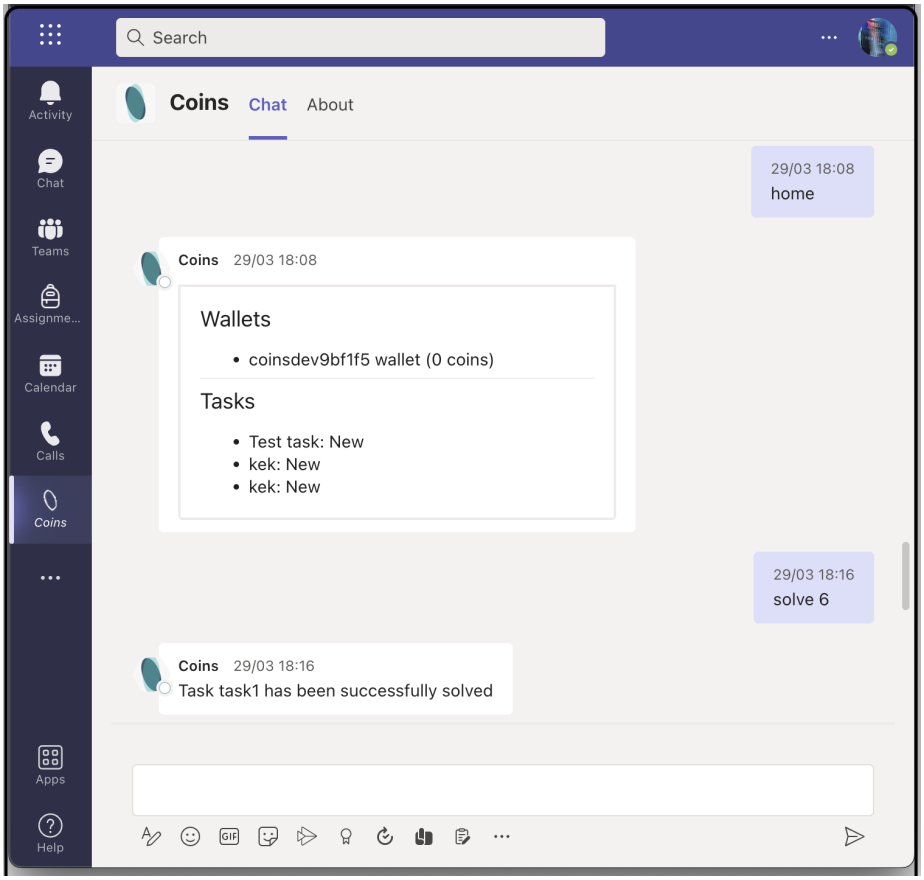
Ponieważ forma studiów zmieniła się w roku 2022, i pojawił się rodzaj zajęć hybrydowych, zarówno studenci, jak i nauczyciele zaczęli korzystać z platformy Microsoft Teams jako głównego narzędzia komunikacji. Pozwala ono tworzyć zespoły dla przedmiotów, umożliwia wymianę wiadomości i prowadzenie lekcji online. Platforma również pozwala udostępniać własne rozszerzenia w sklepie Microsoft Teams.

Rozszerzenie - to jest aplikacja, która może być napisana w różnych językach programowania, dla których twórcy Microsoft Teams udostępnili interfejsy API. Moje rozszerzenie zostało napisane w języku JavaScript. Ścieżka, od napisania kodu, do udostępniania aplikacji wyglądała w następujący sposób (Źródło [11]):

[11] `https://docs.microsoft.com/en-us/microsoftteams/platform` - MS Teams docs

- zainstalować narzędzie Visual Studio Code, albo aplikację terminalową, pozwalającą budować szkielet projektu
- zarejestrować się na platformie Microsoft Azure, na której zostanie uruchomiona aplikacja
- wybrać jeden z rodzaj aplikacji: "Bots", "Tabs", "Message extensions"
- napisać kod aplikacji, który będzie oparty na Microsoft Teams SDK
- uruchomić aplikację na platformie chmurowej Microsoft Azure
- udostępnić uruchomioną aplikację w sklepie Microsoft Teams i dodać ją do zespołu przez administratora

Po wykonaniu tych kroków powstała aplikacja rodzaju "Chat Bot", która pozwalała tworzyć zadania przez członka zespołu i korzystać z funkcjonalności udostępnionej przez serwera.



Rysunek 2.9: Microsoft Teams chatbot

Mimo to, że udało się uruchomić aplikację, z doświadczenia wynika, że koszty implementacyjne, koszty utrzymywania aplikacji i rozwoju, są wyższe, niż koszty budowania aplikacji klienckiej, opartej na framework ReactJS i uruchomionej w przeglądarce. Można wywnioskować, że aplikacja rodzaju "Chat Bot" jest nieporównywalna z aplikacją, która może być utworzona w przeglądarce. Nieporównywalność oznacza, że bot musi służyć na konkretne potrzeby, aplikacja tego rodzaju jest mocno oparta na konwersacjach. Stąd jednym z powodów, dlaczego eksperymenty, związane z rozszerzeniem przestały się rozwijać, jest to, że zarówno dla programisty jak i dla klienta będzie ciężko albo nawet niemożliwe spełnić wszystkie wymagania projektu.

Chatboty są przydatne, gdy istnieje potrzeba w utworzeniu asystentów inteligentnych, albo w robieniu niewielkiej ilości funkcjonalności typu wyszukiwania albo wykonywania szybkich poleceń.

Warte odnotowania jest, rozszerzenia Microsoft Teams również uruchamiają się na frameworku "Electron" (Źródło[12]), który z kolei pozwala budować aplikacje dla urządzeń mobilnych i urządzeń Desktop. Electron jest oparty na technologiach webowych, co z kolei oznacza, że cały kod, zawierający logikę biznesową (oprócz szczególnych przypadków), który został napisany w języku JavaScript dla aplikacji Microsoft Teams może być uruchomiony w przeglądarce.

Innym powodem, dla którego ta aplikacja została odrzucona, jest to, że model biznesowy Microsoft nie pozwala na uruchamiane tej aplikacji na innych platformach oprócz Microsoft Azure. Cała dokumentacja i konfiguracja projektu zakłada, że projekt będzie technologii Azure, z tym pojawiają się kolejne koszty w utrzymywaniu aplikacji i wykrywanie błędów staje się trudniejsze, bo nie ma możliwości uruchomienia zarówno części serwerowej jak i klienckiej na jednej maszynie, albo na jednej przestrzeni chmurowej, takiej jak na przykład Heroku.

2.5.4 Integracja z systemem USOS

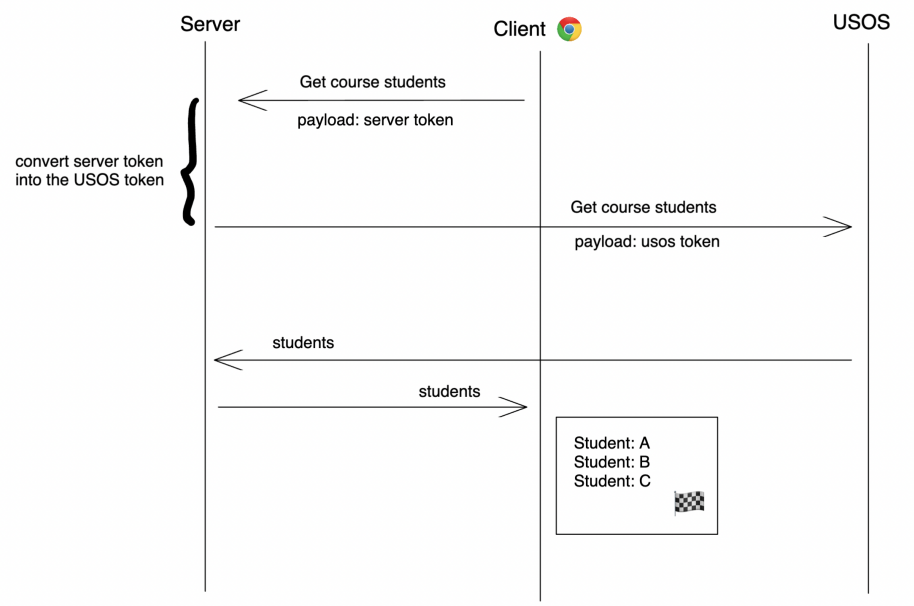
System uniwersytecki "USOS", w którym studenci rejestrują się na przedmioty, gra podstawową rolę ostatecznej postaci. Aplikacja Coins jest połączona z tym systemem, żeby tworzyć nowe pokoje, w których studenci będą wykonywali zadania. System USOS, podobnie jak MS Teams, również uwierzytelnia użytkowników. USOS udostępnia swoją funkcjonalność na podstawie protokołu autoryzacyjnego OAuth 1.0a[13].

Cała komunikacja opiera się na tokenach które pochodzą z wyżej wymienionego protokołu. Student najpierw musi zalogować się do aplikacji Coins przez system USOS, wyrażając zgodę na udostępnianie adresu mailowego i numeru indeksu.

Po zalogowaniu się, tokeny USOS przechowywane są wyłącznie w bazie danych po stronie serwerowej. Natomiast serwer generuje nowy token JWT[14], który

[12] `https://www.wikiwand.com/en/Electron_(software_framework)` - Elektron framework
[13] `https://oauth.net/core/1.0a` - Protokół autoryzacyjny OAuth 1.0a
[14] `https://jwt.io` - Tokeny JWT

zwraca się użytkownikowi w odpowiedzi. Tokeny JWT są potrzebne, żeby autoryzować użytkownika. Żeby wykonać zapytanie na serwer, trzeba wysłać token, który wcześniej był wygenerowany przez serwer po zalogowaniu się przez system USOS. Serwer z kolei, rozpoznaje, kto robi zapytanie i w wyniku zwraca odpowiedź



Rysunek 2.10: Wyszukiwanie studentów, które biorą udział w przedmiocie

Rozdział 3

Wdrożenie aplikacji

3.1 Wdrożenie aplikacji

Ostatecznym krokiem jest udostępnianie aplikacji użytkownikiem i wdrożenie (ang. Deployment) jej na serwisach chmurowych. Wybór serwisu opiera się na takich samych zasadach jak i wybór każdej innej technologii dla projektu, natomiast na potrzeby tego projektu szczególną uwagę przywiązuje się do prostoty sposobu konfiguracji serwisu. Istotne jest na ile taki serwis chmurowy jest popularny i niezawodny.

Skoro podczas projektowania aplikacji zostały uwzględnione wszystkie zasady wcześniej wspomnianych dwunastu reguł "Twelve factor", które uniezależniają architekturę projektu od platformy, na której ten projekt zostanie uruchomiony, odpowiednim miejscem wdrożenia aplikacji wydaje się platforma Heroku. Ta platforma pozwala pozbyć się złożonych konfiguracji serwera, połączyć projekt z bazą danych i uruchomić go na serwerach tej chmury. Dodatkową zaletą jest to, że Heroku rozpoznaje automatycznie rodzaj projektu, na przykład czy to jest projekt platformy JVM z systemem budowania Gradle (projekt backend), czy to jest projekt frontendowy, który zostanie uruchomiony na technologii NodeJs (projekt frontend).

Platforma Heroku w wersji darmowej wydawała się zupełnie wystarczającym rozwiązaniem. Niestety z czasem wynikło, że serwer, uruchomiony dla części frontend nie był w stanie obsługiwać wszystkich żądań użytkowników. Mianowicie problem polegał na tym, że brakowało standardowej dostępnej pamięci RAM i serwer musiał cały czas uruchamiać się od nowa. To jest znany problem Heroku o nazwie "Memory quota vastly exceeded"[1]

Skoro ten problem nie dotyczył części serwerowej, dla części frontend został wy-

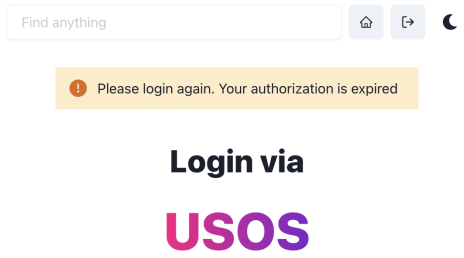
[1] `https://devcenter.heroku.com/articles/error-codes` - kody błędów Heroku

brany inny popularny serwis "Vercel"[2], dedykowany jedynie dla aplikacji frontend, zbudowanych na podstawie najbardziej popularnych bibliotek, takich jak ReactJs. Ten serwis bierze pod uwagę ilość zasobów których potrzebują projekty napisane na podstawie wybranej biblioteki ReactJS i działa stabilnie.

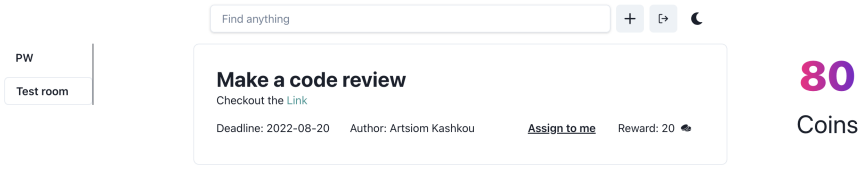
Ostatecznie udało się uruchomić aplikację backend na platformie "Heroku", a aplikację frontend na platformie "Vercel", one komunikują się na postawie skonfigurowanych adresów w zmiennych środowiskowych.

3.2 Interfejs użytkownika

Gdy nowy użytkownik wchodzi do aplikacji, musi uwierzytelnić się w systemie USOS. Po zalogowaniu się, następnym krokiem jest wybranie pokoju. Na poniższych zdjęciach również przedstawiono formularz tworzenia nowego zadania i formularz, odpowiadający za tworzenie pokoju przez administratora. Interfejs użytkownika jest dostępny jedynie w języku angielskim i ma możliwość przełączania się pomiędzy motywem jasnym i ciemnym.

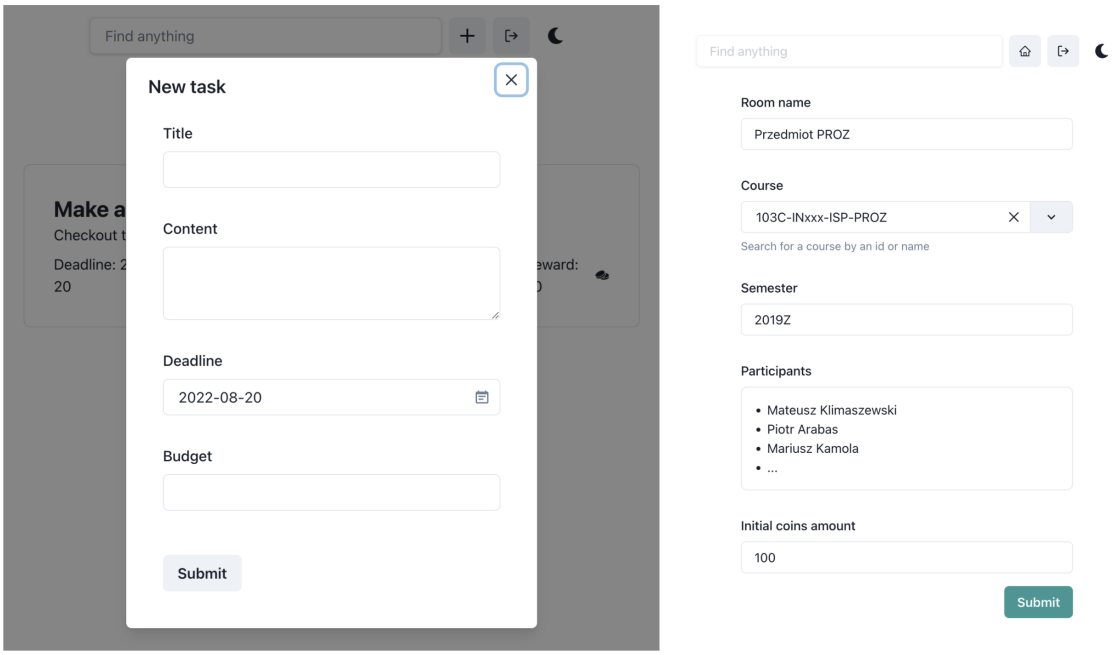


Rysunek 3.1: Ekran logowania



Rysunek 3.2: Ekran domowy

[2] `https://vercel.com` - serwis dla wdrożenia aplikacji frontend



Rysunek 3.3: Formularz tworzenia nowego zadania i nowego pokoju

Rozdział 4

Testowanie

Kiedy pojawia się temat testowania trzeba najpierw określić warstwy, które zostaną przetestowane. Po stronie serwerowej można testować zarówno warstwę bazy danych jak i warstwę serwisową, albo na przykład można pisać testy integracyjne, sprawdzające poprawność działania całej aplikacji.

Testowanie każdej warstwy osobno jest bardzo czasochłonne i skoro projekty są zbudowane na podstawie frameworków i bibliotek, które już zawierają testy, głównym celem w ramach projektu jest skupienie się na poprawności działania aplikacji dla różnych scenariuszy, głównie od strony użytkownika.

Zostały napisane testy integracyjne po stronie serwerowej, które w większości przypadków tworzą zapytania do serwera i porównują wyniki z oczekiwanymi. Główną biblioteką dla testów jest "JUnit 5"[1], będącą standardem de facto w przypadku projektów opartych na frameworku Spring.

Żeby testy poprawnie się wykonywały, trzeba skonfigurować osobną bazę danych, bo wykonywanie poleceń na bazie produkcyjnej jest bardzo ryzykowne, można w ten sposób wyczyścić dane użytkowników. Bazą testową jest osobno utworzona baza PostgreSQL. Podczas pisania testów jedynie trzeba czyścić niezbędne tabele po każdym wykonanym teście. Czyszczenie tabel zapewnia to, że przed każdym testem, baza znajduje się w stanie początkowym, czyli nie zawiera danych i wtedy nie trzeba zastanawiać się jakie dane mogą zepsuć scenariusze testowe.

[1] `https://junit.org` - JUnit 5

Rozdział 5

Podsumowanie

W ramach niniejszej pracy została wykonana aplikacja przeglądarkowa. Jest dostępna dla dowolnego użytkownika, znającego adres strony internetowej[1]. Żeby przetestować pełną funkcjonalność, trzeba mieć konto w systemie uniwersyteckim "USOS" Politechniki Warszawskiej. Główną zaletą aplikacji jest to, że ona udostępnia podstawową funkcjonalność, żeby wdrożyć ją w zespołach, które będą powstawać na różnych przedmiotach. Jednocześnie podczas projektowania aplikacji zostały zebrane dodatkowe pomysły, które pozwoliłyby rozszerzyć funkcjonalność aplikacji. To może być na przykład wprowadzenie dodatkowych paneli administracyjnych dla zbierania statystyk o użytkownikach, rozbudowanie scenariusza rozwiązywania zadań, w których autor może automatycznie przyjmować zadania, gdy testy wykonują się poprawnie i inne.

Ponieważ istnieją jeszcze przypadki użycia, które polepszyłyby działanie aplikacji i do tego architektura aplikacji jest zaprojektowana w taki sposób, że każdy student, który ma dostęp do źródeł, może zaczynać natychmiastowo rozszerzać funkcjonalność aplikacji na swoim komputerze, wtedy w dalszych planach jest udostępnianie kodu aplikacji studentom. Ci studenci, którzy uczą się na przedmiotach, podczas których oni poznają technologie, na których bazuje się ten projekt, mogą na istniejącym projekcie wykonywać zadania, które mogą być zdefiniowane przez prowadzącego, albo realizować własne pomysły w przypadku, gdy dostaną na to zgodę.

W kolejnych krokach aplikacja Coins zostanie zintegrowana z platformą Github, w której studenci zarządzają swoimi projektami i udostępniają kod. Integracja będzie polegała na tym, że zadania, które zostaną utworzone w systemie Github można będzie łączyć z zadaniami z aplikacji Coins. To jest możliwe dzięki temu, że Github udostępnia interfejs (API). Podczas tworzenia zadania w systemie Coins, użytkownik będzie mógł wyszukiwać i dodawać adresy stron Github do opisu zadania.

[1] `https://coins-nu.vercel.app/`

Aplikacja będzie udostępniona studentom stopniowo. Najpierw ona będzie wdrożona w kilku zespołach, składających się z około dwudziestu osób. Żeby serwer i baza danych działały jeszcze bardziej stabilnie i mogły obsługiwać wiele użytkowników, w dalszych planach jest rozmieszczenie aplikacji na serwerach firmy "Jetbrains", która może udostępnić swoje zasoby na potrzeby studentów.

Bibliography

[1] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2003. ISBN: 0321125215.

[2] Herberto Graca. *DDD, Hexagonal, Onion, Clean, CQRS, ... How I put it all together*. URL: `https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together`. (accessed: 07.09.2022).

[3] Vlad Khononov. *Learning Domain-Driven Design*. O'Reilly Media, Inc., 2021. ISBN: 9781098100131.

[4] Adam Wiggins. *Adam Wiggins: Twelve factor*. URL: `https://12factor.net`. (accessed: 07.09.2022).

[5] Irina Yakutenko. *Wola i samokontrola*. Alpina non-ficiton, 2018. ISBN: 978-5-91671-732-7.