Implementacja aplikacji wspierającej naukę do egzaminu teoretycznego na prawo jazdy

(Implementation of an application supporting learning for the driving theory test)

Krystian Jasionek Bartosz Janikowski

Praca inżynierska

Promotor: dr Wiktor Zychla

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

1 września 2023

Streszczenie

Celem pracy jest omówienie przygotowanej aplikacji internetowej w wersji na komputery oraz urządzenia mobilne, pozwalającej użytkownikom przygotować się do części teoretycznej państwowego egzaminu na prawo jazdy. Przeprowadzono przegląd i analizę istniejących na rynku konkurencyjnych aplikacji. Na ich podstawie zaprojektowano i zrealizowano w pełni darmową aplikację o wyróżniającym się stylu wizualnym oraz funkcjonalności odpowiadającej potrzebom użytkowników. Projekt konsultowano i testowano z grupą potencjalnych użytkowników oraz osób, które wcześniej korzystały z konkurencyjnych rozwiązań.

The aim of this work is to present a web application created for both desktop and mobile devices, that allows users to prepare for the theoretical part of the state driving license exam. A review and analysis of competing applications on the market has been conducted. Based on this, a fully free application with a distinctive visual style and functionality tailored to the users' needs was designed and implemented. The project was consulted and tested with a group of potential users and individuals who had previously used competitive solutions.

Spis treści

Rozdział 1.

Wprowadzenie

1.1. Motywacja

Przedmiotem pracy jest aplikacja internetowa przeznaczona na komputery oraz urządzenia mobilne, wspierająca naukę do części teoretycznej państwowego egzaminu na prawo jazdy. Państwowy egzamin na prawo jazdy składa się z pytań dotyczących wiedzy teoretycznej związanej z ruchem drogowym oraz obsługą pojazdów mechanicznych, a także oczekiwanego zachowania kierowcy w częstych sytuacjach na drodze.

Obecnie istnieje wiele aplikacji o podobnym przeznaczeniu, takich które symulują formę egzaminu państwowego, często oferując przy tym dodatkową funkcjonalność wspomagającą proces nauczania użytkownika, np. podręcznik i wykłady wideo czy śledzenie postępów użytkownika w nauce. Większość z tych aplikacji cechuje niekoniecznie atrakcyjny wizualnie styl graficzny i nie zawsze przemyślane interfejsy użytkownika. Bardzo często nawet podstawowe funkcje aplikacji, takie jak wykonywanie próbnych egzaminów, są dostępne za opłatą.

W poniższej pracy przedstawiono w pełni darmową, otwartoźródłową aplikację, której celem jest rozwiązanie najczęstszych problemów konkurencyjnych produktów. Wiele uwagi poświęcono przygotowaniu atrakcyjnego wizualnie i oryginalnego projektu graficznego, nawiązującego do estetyki wyścigów samochodowych z początku XX wieku. Projekt został zrealizowany w wersji na urządzenia stacjonarne oraz mobilne. Zaproponowano również metody wspomagania procesu nauczania oraz śledzenia postępów w nauce, w tym dostępny z poziomu aplikacji podręcznik.

Jako autorzy, ze względów ideowych, uznaliśmy za bardzo istotne, by projekt pozostał niekomercyjny. Zdecydowana większość użytkowników aplikacji tego typu to osoby młode, jeszcze nieposiadające własnego dochodu, dlatego uważamy za zasadne udostępnienie produktu nieodpłatnie. Dodatkowo planujemy udostępnić kod źródłowy ze względu na jego możliwe walory edukacyjne oraz licząc na ewentualną współpracę z gronem użytkowników naszej aplikacji w przyszłości i ich pomoc w

rozbudowie i doskonaleniu projektu.

1.2. Gotowa aplikacja

Repozytorium aplikacji znajduje się pod adresem:

`https://github.com/Sztakler/driver-license-uwr`.

Strona została zahostowana pod adresem:

`https://sztakler.github.io/driver-license-uwr/`.

Możliwe jest także uruchomienie jej lokalnie, w tym celu należy wypełnić plik konfiguracyjny dla bazy danych, czyli `/src/server/files/DatabaseConfiguration/database.js`. Należy ustawić wartości zmiennych (poprawne wartości zmiennych można uzyskać bezpośrednio od autorów pracy) oraz plik konfiguracyjny aplikacji `/src/server/client/configure_build.js`, a konkretnie upewnić się, że wartość zmiennej *urlToServer* wynosi *"http://localhost:3000"*.

Po wykonaniu powyższych kroków należy przejść do katalogu głównego i uruchomić odpowiednie polecenia w terminalu:

```
npm install --legacy-peer-deps
```

aby zainstalować potrzebne moduły oraz

```
npm start
```

w celu uruchomienia aplikacji. Do pełnego funkcjonowania potrzebujemy jeszcze serwera, do tego należy uruchomić osobne okno terminala oraz podać kolejno wymienione polecenia w katalogu `/src/server`:

```
npm install
```

```
npm run dev
```

Domyślnie skrypt umieszcza stronę pod linkiem:

`http://localhost:1234/driver-license-uwr/`.

Po wejściu na nią użytkownik znajduje się na stronie domowej, która reklamuje najważniejsze atuty aplikacji i zachęca do darmowej rejestracji. Wszystkie pozostałe obszary witryny są dostępne dopiero po zalogowaniu, w przeciwnym wypadku odsyłając odwiedzającego do ekranu logowania.

1.3. Budowa projektu

Na aplikację składają się trzy warstwy:

- **frontend** – część użytkowa aplikacji. Zawiera symulację egzaminu, tryb treningu, podręcznik oraz profil użytkownika.
- **backend** – odpowiada za przesył informacji między bazą danych a warstwą użytkownika.
- **baza danych** – przechowuje pytania egzaminacyjne wraz z przypisanymi im mediami oraz dane użytkowników.

Każda z powyższych warstw oraz zastosowane w nich rozwiązania zostaną dokładnie opisane odpowiednio w rozdziałach **Frontend**, **Backend** oraz **Baza danych**.

1.4. Podział prac

Ze względu na rozmiary aplikacji podział prac był koniecznością, jednak jako autorzy traktowaliśmy projekt jako sposób na doskonalenie swoich umiejętności i zdobycie doświadczenia w zakresie tworzenia aplikacji internetowych. Z tych powodów zdecydowaliśmy, że obaj będziemy odpowiadać za przygotowanie zarówno warstwy użytkownika, jak i serwera, a będziemy się dzielić fragmentami kodu lub odpowiedzialnością za implementację poszczególnych komponentów. Stosowaliśmy również metodę programowania w parze, gdzie wspólnie pomagaliśmy sobie w rozwiązaniu problemu.

Do współpracy nad tworzeniem kodu korzystaliśmy z systemu kontroli wersji Git[1] w serwisie Github[2]. W pracy nad interfejsem strony wspomagaliśmy się programem Figma[3], a do robienia wspólnych notatek korzystaliśmy z HackMD[4].

1.5. Plan pracy

W rozdziale drugim przedstawiono analizę zagadnienia budowy aplikacji do wspomagania nauki na część teoretyczną egzaminu na prawo jazdy. Opisano także cele projektu oraz elementy zaimplementowane w finalnej wersji.

W rozdziale trzecim zawarto przegląd i analizę konkurencyjnych aplikacji.

Rozdział czwarty obejmuje przegląd oraz opis rozwiązań i technologii, jakie zastosowano w naszym projekcie, a także pokrótce opisano rozbieżności między zamierzeniami projektu a jego implementacją.

Rozdział piąty zawiera przypadki użycia aplikacji, a także podręcznik użytkownika, który przedstawia przygotowane funkcje aplikacji oraz projekt interfejsu.

Rozdział szósty omawia szczegóły implementacji strony serwerowej, jego architekturę oraz strukturę projektu. Zawiera również prezentację i omówienie modelu bazy danych.

Rozdział siódmy zawiera podsumowanie projektu oraz przedstawia cele i plany autorów w kontekście dalszych prac nad aplikacją.

Rozdział 2.

Opis i analiza zagadnienia

Projekt rozpoczęto z ambitnymi zamierzeniami. Miał na celu rozwiązać problemy konkurencyjnych aplikacji, a ponadto zapewnić użytkownikom zdecydowanie lepsze i bardziej zróżnicowane i rozbudowane metody nauki oraz śledzenia postępów. Planowano:

- utworzyć estetyczny interfejs w oryginalnej stylistyce,
- wprowadzić kilka trybów nauki:
  1. tryb symulacji egzaminu, będący wierną kopią egzaminu państwowego,
  2. tryb nieskończony, w którym kursant odpowiadałby na kolejne pytania bez końca
  3. tryb kategorii, zawierający w puli pytań jedynie pytania z danej tematyki, np. znaki drogowe czy pierwsza pomoc,
- napisać własny lub zredagować istniejący, oparty na wolnej licencji podręcznik, pokrywający wszystkie obszary wiedzy wymagane na egzaminie oraz wzbogacony o ilustracje, infografiki oraz animacje utrzymane w spójnym z resztą interfejsu stylu graficznym,
- dodanie elementów grywalizacji, inspirowanych rozwiązaniami stosowanymi w Duolingo[5].

  Aplikacja śledziłaby postępy użytkowników i przyznawała im punkty za ukończone egzaminy lub treningi, a także bonusy do zdobywanych punktów. Kursanci byliby umieszczani w listach rankingowych oraz ligach, w których konkurowaliby z innymi użytkownikami o najwyższe miejsca. Dodatkowo byliby motywowani do codziennej nauki przez mechanikę *ciągów nauki* (odpowiednik w Duolingo to *streak*). Za opuszczony dzień nauki gracz byłby karany obniżeniem pozycji w rankingu, odebraniem premii do zdobywanych punktów lub przeniesieniem do niższej ligi.

- udostępnić użytkownikom zestaw statystyk i danych dotyczących ich postępów w nauce oraz pomoce naukowe, mianowicie:
  - wykres procentowy przyswojonego materiału,
  - wykres słupkowy liczby egzaminów i treningów ukończonych w ciągu ostatnich dni,
  - listę zapisanych przez użytkownika pytań,
  - listę ukończonych egzaminów i treningów z możliwością podglądu każdego z nich,
  - oznaczenia poziomu znajomości danego pytania i automatyczne metody określania tego poziomu.

Szybko okazało się, że realizacja wszystkich naszych zamierzeń jest niemożliwa ze względu na ograniczenia czasowe i niewielkie rozmiary naszego zespołu. Dlatego skupiliśmy się na realizacji najistotniejszych obszarów strony, by zapewnić jej podstawową funkcjonalność, oraz na pełnej implementacji interfejsu. Na tych fundamentach zamierzamy w przyszłości rozbudowywać funkcjonalność aplikacji o zamierzone wcześniej elementy.

W chwili prezentacji pracy zaimplementowano w pełni:

- interfejs aplikacji,
- tryb symulacji egzaminu,
- tryb treningu z filtrami, umożliwiającymi losowanie spośród pytań wcześniej zapamiętanych przez użytkownika albo oznaczonych przez niego jako dobrze, średnio i słabo znane.

Udało się również przygotować wersję demonstracyjną podręcznika, która ma prezentować jego oprawę wizualną i przykładową zawartość merytoryczną. Zawiera tylko jeden rozdział, dotyczący znaków drogowych.

Zapewniliśmy także podstawową funkcjonalność panelu użytkownika, umożliwiając kursantowi śledzenie statystyk dotyczących nauki i pomocy naukowych:

- procent przyswojonego materiału,
- liczba egzaminów i treningów ukończonych w ostatnich dniach,
- lista zapisanych przez użytkownika pytań.

W miejsce automatycznych metod określania znajomości pytania zdecydowaliśmy się udostępnić użytkownikom jedynie sposób manualny.

Rozdział 3.

Przegląd konkurencyjnych rozwiązań

Obecnie na rynku znajduje się wiele aplikacji, służących do nauki do egzaminu na prawo jazdy, jednak większość z nich cierpi na co najmniej jeden z poniższych problemów:

- nieatrakcyjna szata graficzna i źle przemyślany interfejs,
- wymaganie zapłaty za korzystanie z produktu,
- uboga funkcjonalność wspierająca naukę.

W kolejnych sekcjach zostaną omówione pod kątem wspomnianych problemów dwie popularne aplikacje, które realizują odmienne filozofie. Oczywiście oprócz nich na rynku znajduje się wiele innych rozwiązań, między innymi:

- `https://e-testynaprawojazdy.pl`,
- `https://egzaminword.pl`,
- `https://www.zdamyto.com`,
- `https://testy-na-prawko.pl`,

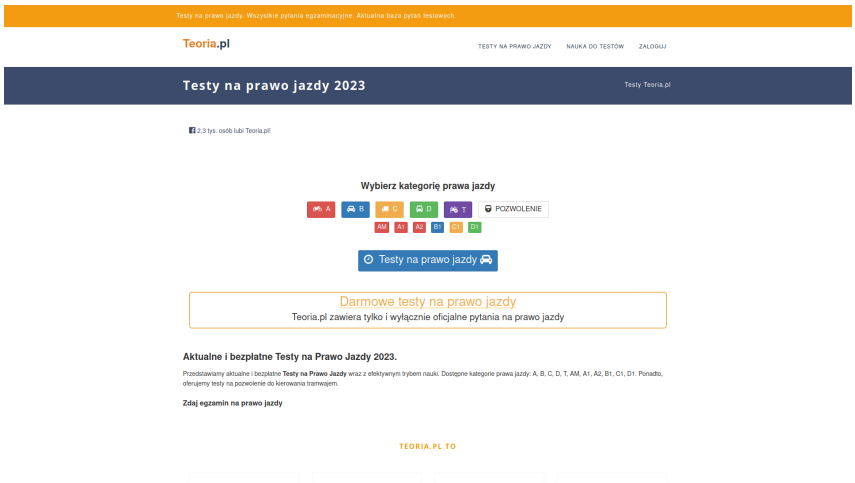
jednak wszystkie one padają ofiarą przytoczonych powyżej problemów. Te dwie konkretne aplikacje wybrano ze względu na to, że uważamy je za najlepszą ilustrację wymienionych wad.

3.1. Witryna teoria.pl

Teoria.pl to przykład aplikacji w pełni darmowej, ale oferującej ubogie doświadczenie użytkownika zarówno pod względem estetycznym, wygody użytkowania, jak i pomocy naukowych. Jest dostępna pod adresem `https://www.teoria.pl/`.

3.1.1. Szata graficzna i interfejs użytkownika

Po wejściu na stronę główną użytkownik widzi bardzo minimalistyczny projekt graficzny (3.1). Białe tło, prostokątne przyciski, łagodne animacje, teksty napisane klasycznym krojem Helvetica, wszystko składa się na bardzo schludny i przejrzysty interfejs.



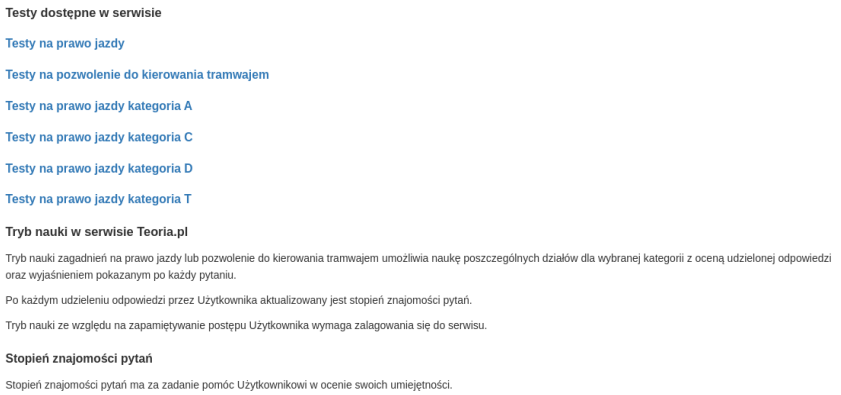
Rysunek 3.1: Strona główna testy.pl

Wrażenie to psuje nagła mnogość kolorów, użytych w menu wyboru kategorii prawa jazdy (3.2). Takie rozwiązanie co prawda przykuwa uwagę, ale jest krzykliwe. Sytuację pogarszają okazjonalne literówki w typografii.

Rysunek 3.2: Bałagan kolorystyczny w menu kategorii

Kolejny element witryny zawiera reklamę wszystkich funkcji aplikacji oraz opisy każdej z nich. O ile oferta została zaprezentowana w minimalistyczny i estetyczny

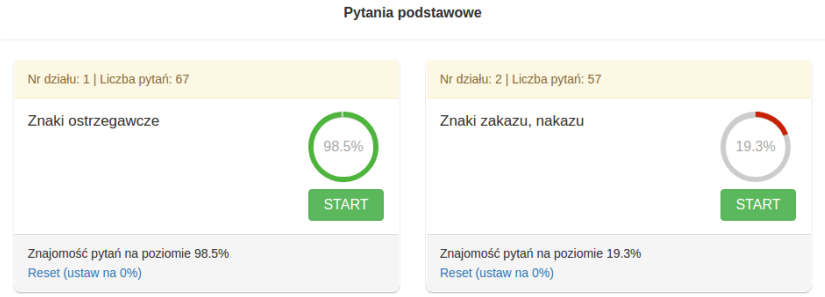
sposób, to zawarty pod nią blok tekstu sprawia dużo gorsze wrażenie 3.3. Zadbano o jego czytelność, zwiększając interlinię, ale sama treść jest przesadnie obszerna, przytłacza użytkownika swoimi rozmiarami i choć zawiera przydatne informacje o działaniu strony i jej zasadach, to niewielkie grono odwiedzających będzie wystarczająco cierpliwe, by się z nimi zapoznać.



Rysunek 3.3: Przesadnie długie opisy funkcji

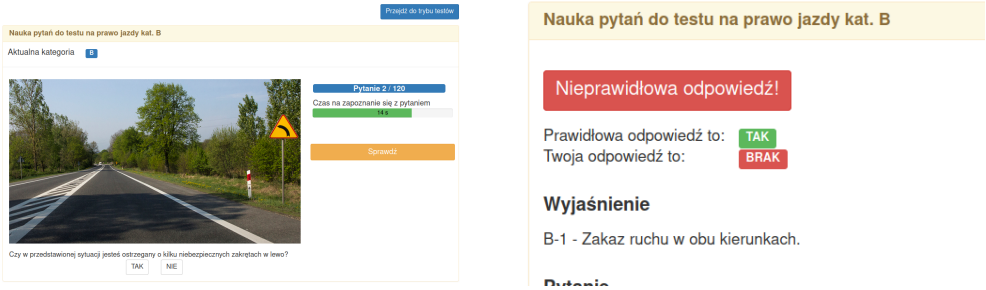
Hierarchia informacji w tym obszarze strony nie zachęca użytkownika do zapoznania się możliwościami aplikacji. Należy pamiętać, że średni czas, jaki internauci spędzają na witrynie, wynosi między dziesięć a dwadzieścia sekund [6], a jeśli chcemy liczyć na ich dłuższą aktywność, musimy przykuć ich uwagę w tak krótkim czasie. W omawianym przypadku prezentacja zawartości jest widoczna dopiero po zjechaniu w dół strony o cały ekran, a wyżej znajduje się jedynie mało atrakcyjne menu kategorii. Nie jest to ustawienie sprzyjające przykuwaniu uwagi odwiedzających stronę.

Tryb treningu został zaprojektowany oszczędnie, ale przejrzyście. Lista działów pytań jest zbyt długa i niełatwo odnaleźć w niej poszukiwaną kategorię (3.4).



Rysunek 3.4: Wybór działu pytań w treningu

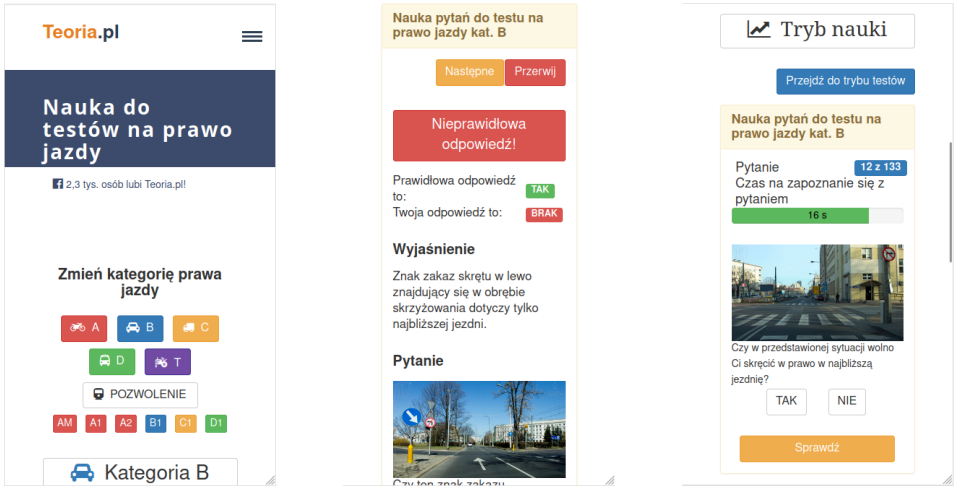
W trybie testów kolorystyka oraz rozmieszczenie elementów odwzorowują interfejs egzaminu państwowego, co może pomóc użytkownikom czuć się pewniej w sali egzaminacyjnej (3.5a). Po wejściu w tryb testu widok jest przesuwany w dół strony, usuwając z pola widzenia nawigację oraz menu kategorii, które mogłyby rozpraszać zdającego. W tym widoku w oczy rzuca się napis *Nieprawidłowa odpowiedź*, która jest ostylowana jak przycisk, ale nie ma przypisanej żadnej akcji (3.5b).



(a) Wybór działu pytań w treningu (b) Nieuzasadnione użycie przycisku

Rysunek 3.5: Widok treningu

Aplikacja jest bardzo słabo przystosowana do wyświetlania na urządzeniach mobilnych. Wydaje się przepełniona zawartością, problemy z centrowaniem oraz brakiem marginesów i odstępów stają się jeszcze bardziej widoczne 3.6.



(a) Strona główna (b) Podsumowanie treningu (c) Trening

Rysunek 3.6: Przykład problemów z mobilną wersją aplikacji

3.1.2. Funkcje aplikacji oraz pomoce naukowe

Aplikacja nie jest zbyt obszerna pod względem funkcjonalności. Udostępnia jedynie dwa tryby nauki:

- egzamin,
- trening.

Pierwszy z nich to symulacja egzaminu państwowego z takimi samymi zasadami i o bliźniaczej oprawie wizualnej. Po ukończonym egzaminie kursant może w podsumowaniu ponownie przejrzeć pytania. Drugi jest dostępny tylko po zalogowaniu i pozwala użytkownikom na wybranie pojedynczego zestawu pytań, np. dotyczących

tylko znaków zakazu i nakazu. W tym trybie widoczny jest też poziom znajomości danej kategorii, mierzony na podstawie liczby pytań, na które odpowiedział użytkownik.

3.1.3. Ogólne wrażenia

Omówiona aplikacja stanowi interesujący przypadek. Z jednej strony stawia na bardzo minimalistyczną stylistykę i przejrzystość interfejsu, z drugiej w wielu miejscach projekt graficzny jest niekonsekwentny lub po prostu nieprzemyślany. Całość, choć z potencjałem na prostą, darmową i funkcjonalną aplikację, sprawia wrażenie wykonanej z małą starannością.

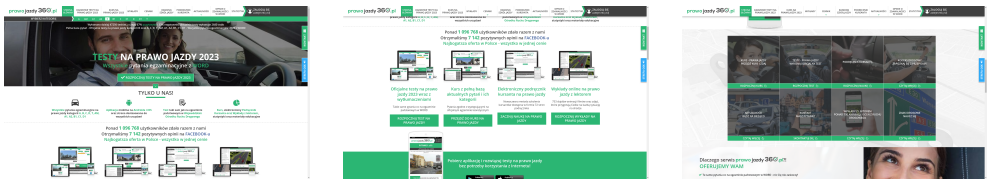
3.2. Witryna prawo-jazdy-360.pl

3.2.1. Szata graficzna i interfejs użytkownika

Ze względu na rozmiary strony nie będziemy omawiać każdego z jej obszarów. Skupimy się na tych najważniejszych dla użytkownika, w szczególności takich, które zaimplementowano w naszej autorskiej aplikacji. Serwis jest dostępny pod adresem: `https://www.prawo-jazdy-360.pl/`.

Strona domowa jest przesycona zarówno pod względem stylistycznym, jak i samej treści. Sam pasek nawigacji zawiera aż dwanaście elementów, z których niektóre dodatkowo rozwijają się w podmenu 3.7.

Trudno odnaleźć się w takim gąszczu informacji. Panuje tu bałagan. Treści marketingowe mieszają się z odnośnikami do konkretnych funkcji aplikacji, w dodatku każdy z odsyłaczy powtarza się wielokrotnie w różnych miejscach na stronie, ale ostylowany na różne sposoby. Raz jest zwykłym przyciskiem, raz kartą z polem tekstowym.

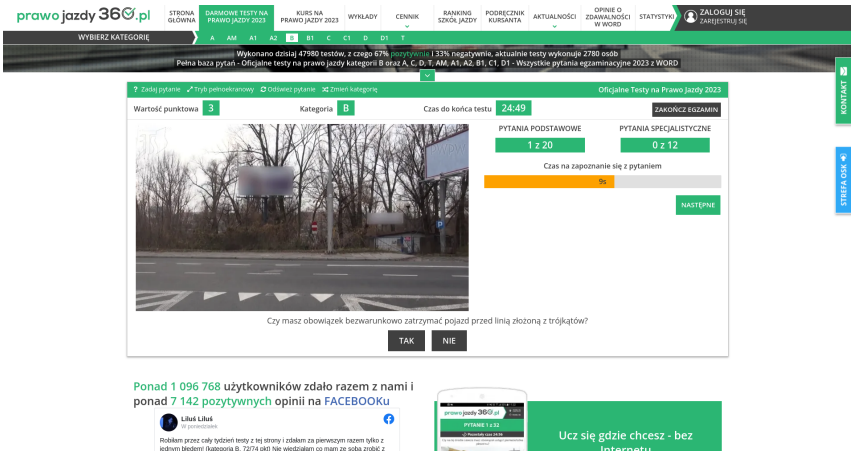


(a) (b) (c)

Rysunek 3.7: Strona główna *Prawo Jazdy 360*

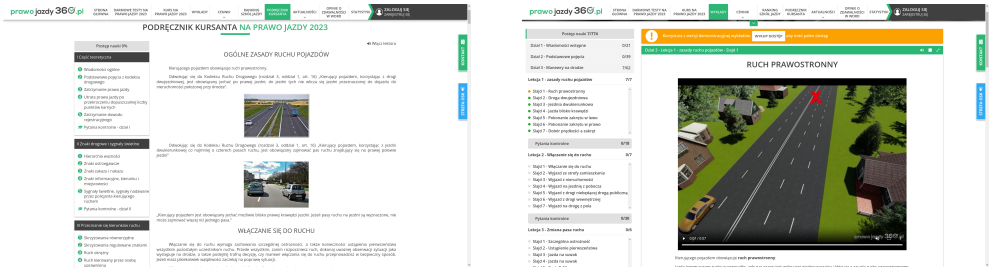
Również w tej aplikacji widok egzaminu jest wzorowany na egzaminie państwowym. Dodano do niego kilka interesujących funkcji, jak możliwość odświeżenia pytania, co resetuje upływ czasu, tryb pełnoekranowy oraz funkcja zadania pytania w wiadomości e-mail do zespołu opiekującego się aplikacją. Sam projekt egzaminu

jest prosty, utrzymany w spójnej z resztą strony stylistyką, a samo wykonanie nie daje powodów do narzekań. Na zakończenie egzaminu zostaje wyświetlony ekran ze statystykami dotyczącymi ukończonego testu, a także lista pytań i prawidłowych odpowiedzi z możliwością wglądu 3.8.



Rysunek 3.8: Widok egzaminu

Widoki wykładów oraz podręcznika łączy bliźniacze podobieństwo. Oba są zorganizowane w schludny i przejrzysty sposób, wspierający przyswajanie treści. Spis treści, pełniący jednocześnie funkcję nawigacji, znajduje się po lewej stronie i pozwala na proste wyszukiwanie oraz przechodzenie między działami. Dodatkowa nawigacja pozwalająca na szybkie przejście do kolejnego lub poprzedniego działu została umieszczona na końcu aktualnego 3.9.



(a) (b)

Rysunek 3.9: Widok podręcznika i wykładu

Podstrona ze statystykami zawiera podstawowe informacje o postępach w nauce: śledzi liczbę egzaminów wykonanych danego dnia, przeczytane rozdziały podręcznika oraz obejrzane wykłady, procent zdanych i niezaliczonych testów, a także lista pytań, z którymi użytkownik miał problem. Można zwrócić uwagę na bardzo duże rozmiary elementów i tekstów. Bez potrzeby zabierają miejsce na stronie i sprawiają, że konieczne staje się przewijanie 3.10.

(a) (b) (c)

Rysunek 3.10: Widok statystyk i zapisanych pytań

3.2.2. Funkcje aplikacji oraz pomoce naukowe

Serwis poza egzaminem oferuje dostęp do podręcznika i wykłady z lektorem. Należy przyznać, że same treści tych obszarów zostały przygotowane starannie i z pewnością są bardzo pomocne w przygotowaniach do egzaminu. Dodatkowym atutem jest mechanizm testu znajomości po każdym rozdziale podręcznika i wykładów. Oferowane statystyki są podstawowe, ale spełniają swoją funkcję.

Brakuje bardziej zaawansowanego trybu nauki, który pozwalałby na przykład przygotowywać egzamin, losując pytania tylko z wybranych kategorii. Obecnie użytkownik może korzystać jedynie z trybu symulacji egzaminu.

Oprócz tego aplikacja oferuje wiele różnych, pomniejszych funkcji, jak przegląd rankingu szkół jazdy, opinie o zdawalności w poszczególnych WORD-ach, a także omówienia częstych pytań zrealizowane w formie artykułów w dziale *Aktualności*.

3.2.3. Ogólne wrażenia

Mimo nieatrakcyjnego i przepełnionego treścią interfejsu serwis jest funkcjonalny. Dostępne w nim metody nauki oraz pomoce prezentują wysoki poziom wykonania. Główną wadą aplikacji jest konieczność opłacania kursu. Bez wydania pieniędzy serwis staje się praktycznie niefunkcjonalny, odsłaniając jedynie kilka obszarów w ramach demonstracji i umożliwiając ukończenie zaledwie dwóch egzaminów.

3.3. Wnioski

Źle zaprojektowana architektura informacji, niedbałość wykonania, przesyt treści to jedne najistotniejszych problemów omówionych serwisów. Pod względem funkcjonalności większość darmowych aplikacji jest bardzo wybrakowana, nierzadko oferując jedynie symulację egzaminu i wyjaśnienia do pytań. Płatne zapewniają więcej, lecz blokują przeważającą część zawartości przed użytkownikiem niezarejestrowanym, przez co w niektórych przypadkach, np. *Prawo Jazdy 360* stają się dla niego zupełnie bezużyteczne.

Analiza tych serwisów pomogła nam w zaprojektowaniu własnej aplikacji, która

zapewniłaby podobną, a w dłuższej perspektywie nawet większą funkcjonalność, przy jednoczesnym uniknięciu ich największych problemów.

Rozdział 4.

Zastosowane rozwiązania

4.1. Projekt interfejsu i szata graficzna

Naszym zamierzeniem było zbudowanie strony atrakcyjnej nie tylko w aspekcie finansowym oraz użytkowym, ale również wizualnym. Estetyczny, oryginalny interfejs stanowił dla nas priorytet. Poprosiliśmy o współpracę zaprzyjaźnioną graficzkę Karolinę Borowską, której zmysł oraz zdolności artystyczne okazały się dla nas nieocenioną pomocą. Karolina po długich sesjach projektowych przygotowała dla nas makietę interfejsu w Figmie, która służyła nam za wzór przy budowie aplikacji. Odpowiada również za wszystkie zawarte w projekcie ilustracje i ikony oraz była pomysłodawcą stylu graficznego aplikacji.

4.2. Hybrydowy model multicloudowy

Do różnych celów w naszym projekcie wykorzystaliśmy różnych dostawców. Nasza baza danych hostowana jest na serwisie *neon.tech*, pliki przechowujemy w *Amazon S3 (buckets)*[7], a serwer hostowany jest przez *Amazon EC2*[8]. Do każdego z celów wybraliśmy innego, najbardziej odpowiadającego nam dostawcę. Nasz model jest przykładem hybrydowego podejścia do korzystania z wielu chmur.

4.3. Frontend

Projekt napisano w języku JavaScript, korzystając z frameworka React[9]. Do stylizacji strony wybrano technologię Tailwind CSS[10], którą rozszerzyliśmy o rozwiązanie *Styled Components*[11]. W poniższych sekcjach opisano pokrótce każdą z technologii.

4.3.1. React

React.js to otwartoźródłowa biblioteka języka JavaScript wykorzystywana do budowy interfejsów. Pozwala na tworzenie reużywalnych komponentów, z których następnie układana jest zawartość aplikacji. Biblioteka wykorzystuje wirtualny DOM, który pozwala na wydajne aktualizowanie zawartości interfejsu w odpowiedzi na zmiany danych.

Biblioteka wykorzystuje składnię deklaratywną, pozwalając programistom w łatwy sposób uzależnić wyświetlaną zawartość od stanu aplikacji. Umożliwia też przepływ danych pomiędzy komponentami, co upraszcza programowanie logiki oraz ułatwia wykrywanie błędów.

Popularność tego rozwiązania przyczyniła się do dynamicznego rozwoju dodatkowych narzędzi, taki jak *React Router*[12], służący do obsługi trasowania, czy *Redux*[13] do zarządzania stanem aplikacji.

W naszej aplikacji użyliśmy *React router* do zarządzania routowaniem (przyporządkowaniem różnym widokom ich wirtualnych *ścieżek*). Do zarządzania współdzieleniem stanu między komponentami skorzystaliśmy z kontekstów [14].

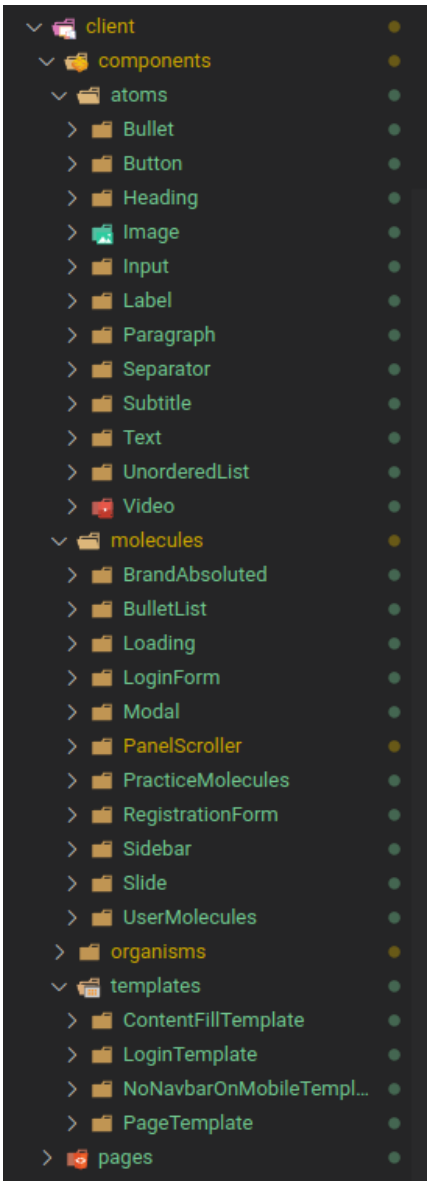
4.3.2. Atomowa architektura komponentów

Od początku pracy naszym celem było logiczne rozdzielenie komponentów w strukturze projektu, aby ułatwić nawigację po nim oraz reużywalność zasobów. Z tego powodu zdecydowaliśmy się wprowadzić *atomową architekturę komponentów* [15]. To rozwiązanie jest źródłem wielu korzyści:

- Modularność – dzięki podziałowi interfejsu na małe komponenty możemy łatwiej zarządzać kodem i wprowadzać w nim zmiany.
- Reużywalność – komponenty niższego poziomu są wielokrotnie wykorzystywane, co przyspiesza rozwijanie nowych funkcjonalności.
- Czytelność – struktura projektu jest bardzo uporządkowana i łatwa do nawigacji.
- Konsystencja – aplikacja jest spójna wizualnie i funkcjonalnie przy odpowiednim podejściu i przestrzeganiu reguł atomowej struktury komponentów.

Przykłady zastosowania

W listingach przedstawiono użycie atomowej architektury komponentów. Do prezentacji wybrano organizm Hero 1 oraz molekułę Slide 2, używane na stronie głównej aplikacji.



Rysunek 4.1: Rozkład folderów w strukturze projektu

```
export default function Hero() {
return (
    <HeroContainer id="HeroContainer">
        <Slide
            id={0}
            bgImage={HomePageIllustrations.HeroImage1_Desktop}
            scrollDown={scrollDown}
        >
        </Slide>
        <Slide
            id={1}
            bgImage={HomePageIllustrations.HeroImage2_Desktop}
            scrollDown={scrollDown}
        >
        </Slide>
        <Slide
            id={2}
            bgImage={HomePageIllustrations.HeroImage4_Desktop}
            scrollDown={scrollDown}
        >
        </Slide>
        <Slide
            id={3}
            bgImage={HomePageIllustrations.HeroImage3_Desktop}
            scrollDown={scrollDown}
            isLastSlide={true}
        >
        </Slide>
</HeroContainer>);
```

Przykładowy kod 1: Organizm - Hero

```
export default function Slide({
id,
scrollDown,
children,
bgImage,
isLastSlide = false,
}) {
return (
    <SlideContainer id={"panel-" + id}>
        <ImageContainer>
            <Image className="w-full h-full max-w-full block" src={bgImage}></Image>
        </ImageContainer>
        {children}
        {!isLastSlide && (
            <ArrowDown>
                <Button onClick={() => scrollDown(id + 1)}>
                    <Image
                        className="flex self-center w-[2vw] "
                        src={HomePageIllustrations.ArrowDown}
                    />
                </Button>
            </ArrowDown>
        )}
    </SlideContainer>
);
}
```

Przykładowy kod 2: Molekuła - Slide

4.3.3. Tailwind CSS

Tailwind CSS to narzędzie do tworzenia interfejsów użytkownika. Opiera się na klasach CSS, co pozwala na szybkie dostosowanie wyglądu elementów interfejsu. Tailwind dostarcza gotowe klasy, ułatwiając projektowanie i zapewniając jednocześnie elastyczność do tworzenia niestandardowych stylów, co znacznie przyspieszyło naszą pracę i umożliwiło skoncentrowanie się na budowaniu interfejsu aplikacji.

Istotne dla nas było, że *Tailwind CSS* oferuje gotowe klasy do projektowania responsywnego. Znacznie ułatwiło to pracę nad mobilną wersją aplikacji.

Motywacja

W trakcie poszukiwań idealnego rozwiązania wypróbowaliśmy kilka z obecnie najpopularniejszych narzędzi do umieszczania stylów na stronie. Z początku w wyborze kierowaliśmy się osobistymi preferencjami, operując w obrębie znanych nam metod, zauważyliśmy jednak, że żadne z tych narzędzi nie spełnia naszych wymagań.

W miarę rozbudowy projektu zaczęliśmy dostrzegać konieczność ustalenia pewnych priorytetów. Najistotniejsze było zapewnienie modularności komponentów, tj. każdy komponent aplikacji powinien mieć przypisane do siebie lokalne style, niewidoczne z poziomu innych komponentów. Duży nacisk położyliśmy również na rozdzieleniu części logicznej strony (JS i HTML) od stylistycznej (CSS). Chcieliśmy w ten sposób zapewnić czytelność kodu i zwiększyć przenośność stylów między komponentami.

Przetestowaliśmy trzy rozwiązania, które zostaną pokrótce omówione w kolejności chronologicznej:

1. **własne klasy CSS** – to podejście oferowało największą swobodę tworzenia i było narzędziem, z którym obaj byliśmy dobrze zaznajomieni. Niestety, w projekcie często pojawiały się problemy. Wiele komponentów aplikacji ma podobną strukturę, przez co często klasy w różnych komponentach otrzymywały te same nazwy, co powodowało liczne konflikty. Rozwiązaniem mogłoby być zbudowanie własnej bazy małych klas CSS, których używalibyśmy do nadawania poszczególnych własności stylizowanym komponentom, np. klasa *bold* pogrubiałaby tekst, a *bg-red* ustawiała tło elementu na czerwony. Odrzuciliśmy jednak ten pomysł jako zbyt czasochłonny. Postanowiliśmy spróbować rozwiązać problemy, wykorzystując inne gotowe rozwiązanie.

2. **moduły CSS** – to pliki CSS o lokalnym zasięgu klas[16]. Ich widoczność jest określana przez strukturę plików, tj. dwa pliki ze stylami umieszczone w różnych katalogach nie będą ze sobą w konflikcie. Dzięki temu pozbyliśmy się wcześniejszych problemów, a dodatkowo zwiększyliśmy czytelność i modularność kodu. Każdy komponent znajdował się w osobnym katalogu razem z

przypisanym do niego plikiem CSS. Niestety, nadal istniała konieczność pisania wszystkich klas od zera. W dodatku moduły CSS zmniejszały czytelność wygenerowanego kodu HTML, dodając do nazw klas unikatowe identyfikatory.

3. **Tailwind CSS** – po zapoznaniu się z popularnymi rozwiązaniami stylizacji, to jedno wydało nam się szczególnie innowacyjne i wygodne. Przypomina pomysł z budową własnej bazy niewielkich klas CSS, który rozważaliśmy na początku projektu. Oferuje gotowy zestaw klas, nadających elementom pojedyncze właściwości, np. *bg-red-500* ustawia kolor tła na czerwony, a *justify-center* centruje zawartość elementu *flex*. Zapewniło to wystarczającą, jak na nasze potrzeby, elastyczność, a jednocześnie pozwalało na powtórne użycie klas w dowolnie wielu komponentach. Kosztem tej wygody była utrata podziału kodu na część logiczną i style, ponieważ Tailwind CSS zakłada, że klasy będą umieszczane bezpośrednio w komponentach HTML. Powodowało to problemy z powtarzającymi się fragmentami kodu, podobnie jak w przykładzie 4.2. Udało nam się ich w dużym stopniu uniknąć, dzięki wykorzystaniu *stylizowanych komponentów*, które zostaną omówione niżej.

```
<div>
  <div class="flex items-center space-x-2 text-base">
    <h4 class="font-semibold text-slate-900">Contributors</h4>
    <span class="rounded-full bg-slate-100 px-2 py-1 text-xs font-semibold
  </div>
  <div class="mt-3 flex -space-x-2 overflow-hidden">
    <img class="inline-block h-12 w-12 rounded-full ring-2 ring-white" src=
    <img class="inline-block h-12 w-12 rounded-full ring-2 ring-white" src=
    <img class="inline-block h-12 w-12 rounded-full ring-2 ring-white" src=
    <img class="inline-block h-12 w-12 rounded-full ring-2 ring-white" src=
    <img class="inline-block h-12 w-12 rounded-full ring-2 ring-white" src=
  </div>
  <div class="mt-3 text-sm font-medium">
    <a href="/" class="text-blue-500">+ 198 others</a>
  </div>
</div>
```

Rysunek 4.2: Przykład powtarzających się klas w Tailwind CSS

Stylizowane komponenty

Styled Components[17] to popularna biblioteka do stylizacji komponentów w React przy użyciu podejścia *CSS wewnątrz JS*. Pozwala ona na definiowanie stylów bezpośrednio w komponentach za pomocą JavaScriptu. Uznaliśmy to rozwiązanie za bardzo interesujące, ponieważ dawało możliwość dynamicznej stylizacji za pomocą propsów lub innych warunków (przykład: 4.3) oraz enkapsulację, ponieważ style zostają dostarczone razem z komponentem.

Zaczęliśmy poszukiwać sposobu na połączenie tego rozwiązania z Talwind CSS i szybko natrafiliśmy na bibliotekę *Tailwind Syled Components*[11]. Jest to rozwiązanie bardzo młode, rozwijane od 2022 roku, ale które zdążyło już zgromadzić spore grono

```jsx
import React from 'react';
import styled from 'styled-components';

// Tworzymy komponent <Title>, który renderuje
// element < h1 > z odpowiednimi stylami
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: blue;
`;

// Tworzymy komponent <Wrapper>, który renderuje element <section>
// z dodanym paddigiem i kolorem tła
const Wrapper = styled.section`
  padding: 4em;
  background: red;
`;

// Gotowych komponentów używamy jak zwyczajnych komponentów reactowych
<Wrapper>
  <Title>To jest stylizowany komponent</Title>
</Wrapper>
```

Rysunek 4.3: Przykład użycia stylizowanych komponentów w React

użytkowników. Jego użycie jest bliźniaczo podobne do stylizowanych komponentów w React, ale pozwala na stosowanie klas z Tailwinda (przykład: 4.4).

```jsx
const Button = tw.div`
    ${(p) => (p.$primary ? "bg-indigo-600" : "bg-indigo-300")}

    flex
    inline-flex
    items-center
    border
    border-transparent
    text-xs
    font-medium
    rounded
    shadow-sm
    text-white

    hover:bg-indigo-700
    focus:outline-none
`
```

Rysunek 4.4: Przykład użycia stylizowanych komponentów Tailwind

W ten sposób udało nam się uzyskać modularność i enkapsulację, jednocześnie oddzielając części kodu dotyczące logiki od zawierąjacych style. W tym celu każdy komponent umieściliśmy w osobnym katalogu, a wewnątrz niego znalazły się dwa pliki: index.js oraz styles.js (4.5b). Pierwszy z nich zawiera zawartość oraz logikę komponentu, drugi style, które są importowane do pliku index.js (4.5a).

```
export {
    HeaderContainer,
    TitleContainer,
    IllustrationContainer,
    Title,
    Subtitle,
    Buttons,
};
```

(a) Przykład eksportu stylów (plik styles.js)

```
import {
    HeaderContainer,
    TitleContainer,
    IllustrationContainer,
    Title,
    Buttons,
} from "./styles";
```

(b) Przykład importu stylów (plik index.js)

Rysunek 4.5: Przykład importu i eksportu stylów komponentu (przykład pochodzi z organizmu Header)

4.3.4. Wersja mobilna aplikacji

Jako technologię frontendu dla mobilnej wersji aplikacji wybraliśmy PWA[18] (Progressive Web App). To nowoczesne rozwiązanie w technologiach webowych, które łączy najlepsze cechy aplikacji webowych oraz mobilnych. Pozwala tworzyć szybkie, responsywne aplikacje, zapewniające komfort użytkowania niezależnie od stanu łącza użytkownika.

PWA wykorzystuje serwisy pracujące w tle (ang. *service workers*), tj. działające w tle aplikacji skrypty, umożliwiające buforowanie offline, wyświetlanie powiadomień push oraz poprawiające wydajność aplikacji. Takie rozwiązanie pozwala na szybkie ładowanie PWA oraz zapewnia ich dostępność nawet w wypadku niestabilnego lub niedostępnego połączenia internetowego.

Kolejną zaletą PWA jest fakt, że nie wymagają one tworzenia osobnej bazy kodu dla różnych platform docelowych i wykorzystują technologie webowe. Zapewniana przez nie prostota implementacji oraz wysoka jakość doświadczenia użytkownika, były głównymi powodami, które skłoniły na do wyboru tego rozwiązania.

4.4. Backend

4.4.1. Node.js

Chcieliśmy postawić na sprawdzone środowisko uruchomieniowe JavaScript po stronie serwera, stąd wybór *Node.js*[19]. Atutem tego narzędzia jest efektywna obsługa równoległych żądań, co w rozwoju naszej aplikacji może okazać się kluczowe. Kolejny atut to duża dostępność platform hostingowych.

4.4.2. Express.js

Nasze zaplecze serwerowe od początku nie miało być zbytnio złożone, stąd chcieliśmy postawić na rozwiązanie, które będzie w stanie w prosty sposób spełnić nasze oczekiwania i nie będzie utrudniać rozszerzania aplikacji w przyszłości. Zdecydowaliśmy się w takim razie na minimalistyczny, lecz efektywny framework do *Node.js* *Express.js*[20].

4.4.3. PostgreSQL

Wybór systemu zarządzania bazą danych był bardzo istotną częścią naszego projektu. Potrzebowaliśmy narzędzia, które będzie wydajne, niezawodne oraz przede wszystkim skalowalne. Wybór padł na *PostgreSQL*[21], narzędzie wcześniej przez nas niewykorzystywane, lecz w pełni odpowiadające naszym potrzebom.

4.4.4. Passport

Przy wyborze biblioteki do autentykacji kierowaliśmy się podobnym myśleniem co przy wyborze *Express.js*, chcieliśmy rozwiązania, które w prosty sposób rozwiąże nasze problemy oraz będzie ją łatwo dostosować do potrzeb projektu. *Passport.js*[22] okazał się świetnym narzędziem do odegrania kluczowej roli w procesie uwierzytelniania.

4.5. Implementacja projektu, a rzeczywistość

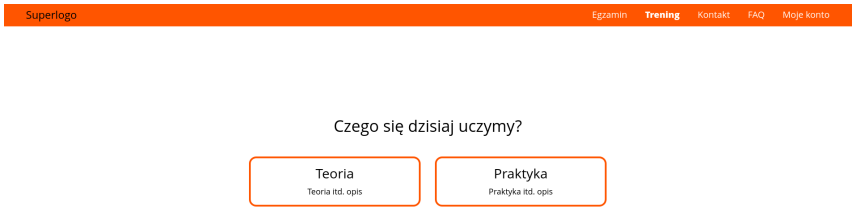
W czasie prac projekt wielokrotnie ewoluował i zmieniał swój kształt. Niektóre zamierzenia okazały się niemożliwe do realizacji w wyznaczonym czasie, inne już po implementacji zostawały zwrócone do poprawek lub zupełnego przebudowania, ponieważ pojawił się nowy, ciekawszy pomysł.

Przykładem takiej sytuacji jest widok egzaminu. Z początku miał być zbliżony do stylistyki egzaminu państwowego (białe tło, prostokątne przyciski w jaskrawych

kolorach, czarny tekst, maksymalizowanie treści i minimalizacja formy), nazwano go nawet *symulacją egzaminu*, jednak gdy tylko został zaimplementowany, stało się jasne, że nie można go pozostawić w projekcie. Zbytnio odstawał stylistycznie od reszty widoków, w czasie testów zarówno my, jak i poproszeni o pomoc testerzy odnosili wrażenie wyrwania z doświadczenia. Z tego powodu cały ten obszar musiał przejść gruntowną przebudowę.

Wiele zmian dotknęło też projekt graficzny. Z racji, że nie posiadaliśmy jasno ustalonych zamierzeń i ani wiedzy odnośnie do budowania interfejsów, to właśnie ten element aplikacji zmieniał się najbardziej dynamicznie. Można wyróżnić w nim trzy okresy:

- **pierwotny projekt** – bardzo uproszczona stylistyka, mniej dopracowane ilustracje, słabo dopasowane kolory. Z tego okresu niewiele zachowało się do wersji ostatecznej poza rozmieszczeniem elementów oraz ogólnym zamysłem na projekt 4.6.

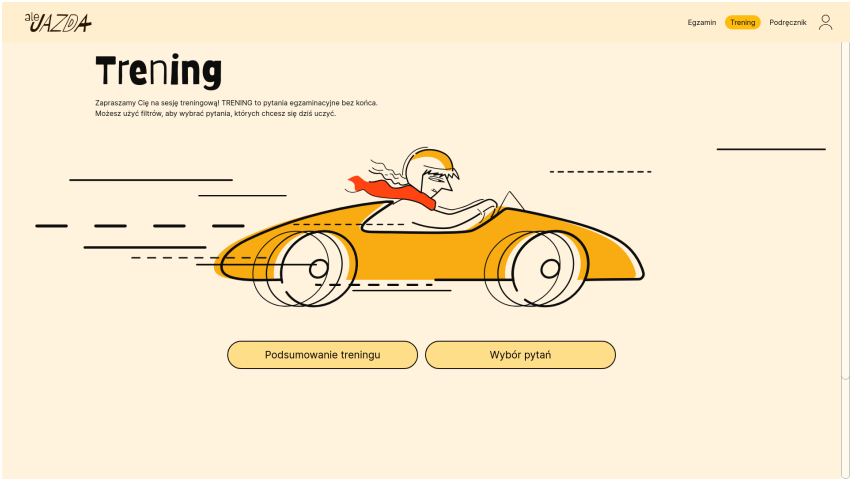


Rysunek 4.6: Pierwsza wersja menu Treningu

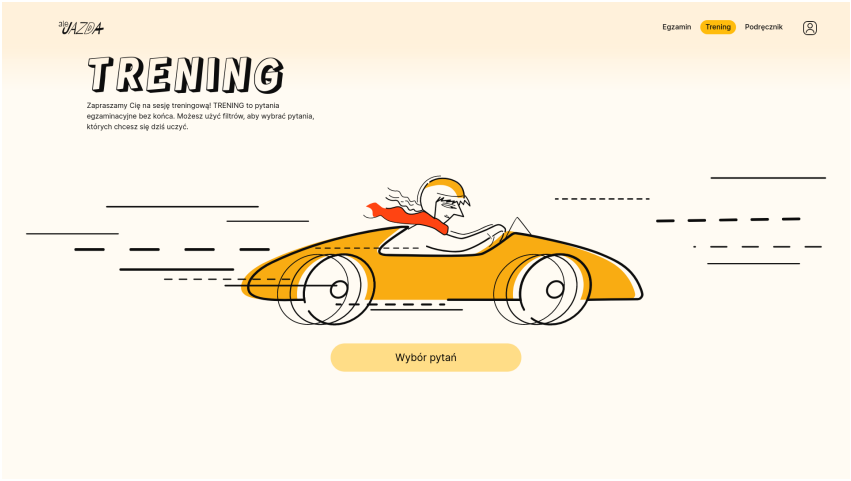
- **usprawniony interfejs** – początkowe próby przygotowania interfejsu z perspektywy czasu trudno opisać inaczej niż jako *niezgrabne*. Architektura informacji na stronie pozostawiała wiele do życzenia, a styl graficzny, choć sama jego idea wydawała się bardzo kreatywna, w rzeczywistości nie był tak przyjemny dla oka. Zdecydowaliśmy się na gruntowną przebudowę interfejsu całej aplikacji, co pochłonęło sporo czasu, ale efekty okazały się warte wysiłku. Powstały projekt jest niemal identyczny z obecnym pod względem architektury informacji, kolorystyką, typografią oraz ilustracjami 4.7.

- **ostatnie szlify** – na tym etapie warstwa techniczna interfejsu znajdowała się w zadowalającym stanie. Postanowiliśmy również zrezygnować ze stylizacji widoku egzaminu na wzór egzaminu państwowego. W przypływie kreatywności zdecydowaliśmy się unowocześnić projekt, dodając do niego gradienty, rozjaśniając kolorystykę oraz usuwając ramki z przycisków i niektórych kontenerów. Finalny projekt stał się zauważalnie lżejszy i przyjemniejszy dla oka 4.8.

Nauczką na przyszłość stało się też dla nas projektowanie mobilnej wersji aplikacji. Skupiliśmy większość środków na tworzeniu implementacji na urządzenia sta-



Rysunek 4.7: Druga wersja menu Treningu



Rysunek 4.8: Obecna wersja menu Treningu

cjonarne, uznając, że budowa wersji mobilnej będzie polegała na modyfikacji kilku plików CSS. Gdy dotarliśmy do tej fazy implementacji, okazało się, że budowa niektórych komponentów, np. Egzaminu oraz Treningu, jest zbyt nieelastyczna i dostosowanie jej pod urządzenia przenośne wymagałoby przepisania całego komponentu niemal od zera. Udało nam się na szczęście znaleźć prostsze, choć nieco mniej eleganstsze rozwiązanie, polegające na łączeniu *media queries* na poziomie CSS oraz kodu JS do podmieniania stylów oraz logiki komponentu między wersjami. Nie było to idealne wyjście z sytuacji, ale akceptowalne i wymagające stosunkowo niewielkich nakładów pracy.

Rozdział 5.

Część dla użytkownika

5.1. Przypadki użycia

Zakładamy (chyba że stwierdzono inaczej), że wszystkie przypadki użycia zaczynają się od otworzenia aplikacji na stronie startowej.

5.1.1. Rejestracja z poziomu strony głównej

1. Użytkownik widzi stronę główną z grafikami promocyjnymi oraz przyciski CTA (*ang. call to action*), zachęcające do rejestracji.
2. Użytkownik klika przycisk CTA i zostaje przeniesiony do strony rejestracji.
3. Użytkownik rejestruje się do aplikacji.

5.1.2. Rejestracja po próbie dostania się do części zastrzeżonej dla zalogowanych użytkowników

1. Użytkownik nie jest zarejestrowany.
2. Użytkownik próbuje dostać się na stronę zastrzeżoną dla zalogowanych użytkowników.
3. Aplikacja przekierowuje użytkownika do ekranu logowania.
4. Użytkownik klika w przycisk *Nie masz jeszcze konta? Zarejestruj się!*.
5. Aplikacja przenosi użytkownika do strony rejestracji.
6. Użytkownik rejestruje się do aplikacji.

5.1.3. Logowanie

1. Użytkownik nie jest zalogowany.
2. Użytkownik próbuje dostać się na stronę zastrzeżoną dla zalogowanych użytkowników.
3. Aplikacja przekierowuje użytkownika do ekranu logowania.
4. Użytkownik loguje się na swoje konto.

5.1.4. Wejście na stronę symulacji egzaminu

1. Użytkownik jest zalogowany.
2. Użytkownik klika w przycisk *Egzamin*.
3. Aplikacja wyświetla stronę z menu egzaminu.
4. Użytkownik klika *Rozpocznij egzamin*.
5. Aplikacja otwiera symulację egzaminu.

5.1.5. Wejście na stronę podsumowania egzaminu

1. Użytkownik jest zalogowany.
2. Użytkownik klika w przycisk *Egzamin*.
3. Aplikacja wyświetla stronę z menu egzaminu.
4. Użytkownik klika *Rozpocznij egzamin*.
5. Aplikacja otwiera symulację egzaminu.
6. Użytkownik rozwiązuje egzamin.
7. a) Użytkownik dotarł do końca egzaminu. b) Użytkownik kliknął *Zakończ egzamin*:
   - Aplikacja wyświetla okno modalne z pytaniem o potwierdzenie zakończenia egzaminu.
   - Użytkownik klika *Tak*.
8. Aplikacja wyświetla widok z podsumowaniem egzaminu.
9. a) Użytkownik kliknął *Nowy egzamin*:
   - Aplikacja wyświetla menu egzaminu.

   b) Użytkownik kliknął *Przejrzyj odpowiedzi*:
   - Aplikacja wyświetla podgląd odpowiedzi

5.1.6. Przeglądanie ukończonego egzaminu

Zakładamy, że użytkownik jest zalogowany i przeszedł na stronę podsumowania.

1. Aplikacja wyświetla podgląd pierwszego pytania w egzaminie.
2. Użytkownik klika *Następne pytanie*.
3. Aplikacja wyświetla podgląd następnego pytania.
4. Użytkownik klika *Poprzednie pytanie*.
5. Aplikacja wyświetla podgląd poprzedniego pytania.
6. Użytkownik klika *Pokaż wyjaśnienie*.
7. Aplikacja wyświetla okno modalne z wyjaśnieniem odpowiedzi na pytanie egzaminacyjne.
8. Użytkownik klika przycisk z krzyżykiem.
9. Aplikacja zamyka okno modalne.
10. Użytkownik klika *Zamknij podsumowanie*.
11. Aplikacja wyświetla okno modalne z prośbą o potwierdzenie zamknięcia podsumowania.
12. Użytkownik klika *Tak*.
13. Aplikacja przenosi użytkownika do menu egzaminu.

5.1.7. Zmiana danych użytkownika (bez podania hasła)

1. Użytkownik jest zalogowany.
2. Użytkownik klika w ikonę użytkownika w nawigacji.
3. Aplikacja przenosi go na stronę użytkownika i wyświetla podstronę *Statystyki*.
4. Użytkownik klika *Ustawienia konta*.
5. Aplikacja przełącza widok na podstronę *Ustawienia konta*.
6. Użytkownik wpisuje nowe dane użytkownika, ale nie wprowadza hasła użytkownika.
7. Użytkownik klika *Zatwierdź zmiany*.
8. Aplikacja podświetla na czerwono pole *Wprowadź aktualne hasło* i wyświetla informację o konieczności podania hasła.
9. Użytkownik wpisuje hasło i klika *Zatwierdź zmiany*.
10. Aplikacja aktualizuje dane użytkownika.

5.2. Podręcznik użytkownika

5.2.1. Ogólne informacje

W kolejnych podsekcjach zostaną omówione wybrane widoki aplikacji oraz ich funkcje. Ze względu na podobieństwo niektórych widoków, niektóre z nich zostaną omówione zbiorczo, na przykład menu treningu, egzaminu oraz podręcznika. W tych przypadkach zostanie to podkreślone na początku podsekcji.

5.2.2. Widok strony głównej

W wersji na urządzenia stacjonarne widok został podzielony na cztery bliźniacze sekcje. Każda z nich składa się z ilustracji, tekstu przedstawiającego możliwości aplikacji oraz przycisku, zachęcającego nowych użytkowników do rejestracji.

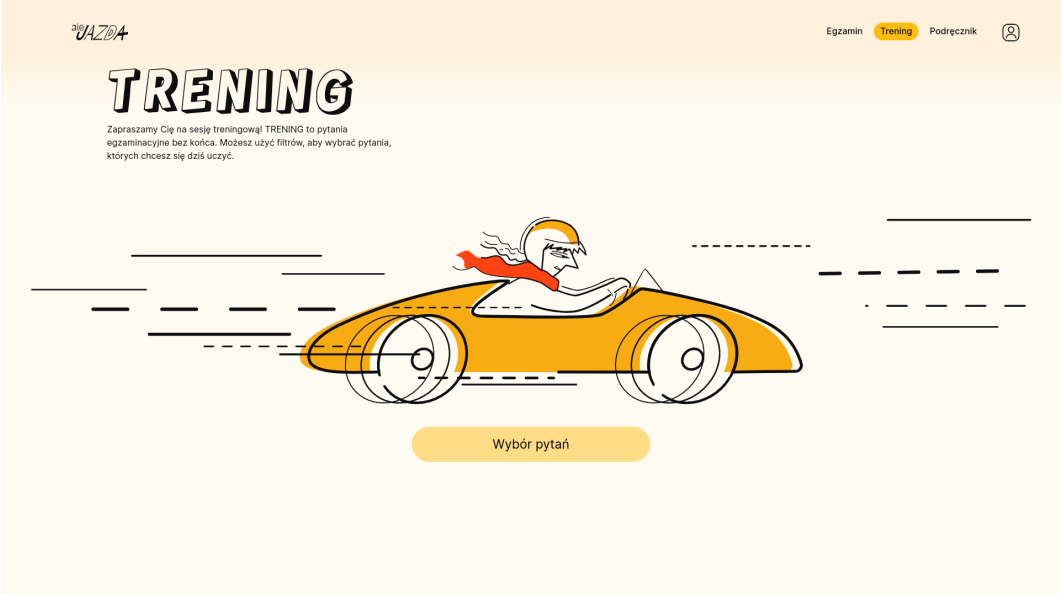
Rysunek 5.1: Wersja na komputery

Na urządzeniach mobilnych aplikacja wyświetla pojedynczy panel z ilustracją i *Call to Action*. Przycisk *Trenuj z nami* jest odsyłaczem do strony rejestracji.

Rysunek 5.2: Wersja na urządzenia mobilne

5.2.3. Widok menu

Strony Egzaminu, Treningu oraz Podręcznika zaczynają się od menu, zawierającego ilustrację, tekst z wyjaśnieniem przeznaczenia i krótką instrukcją obsługi danej strony, oraz przycisk, odsyłający do funkcjonalnej zawartości tego obszaru aplikacji. Rysunki 5.3 i 5.4 przedstawiają odpowiednio stacjonarną i mobilną wersję menu Treningu. W jego przypadku przycisk *Wybór pytań* przenosi użytkownika do widoku filtrów pytań.

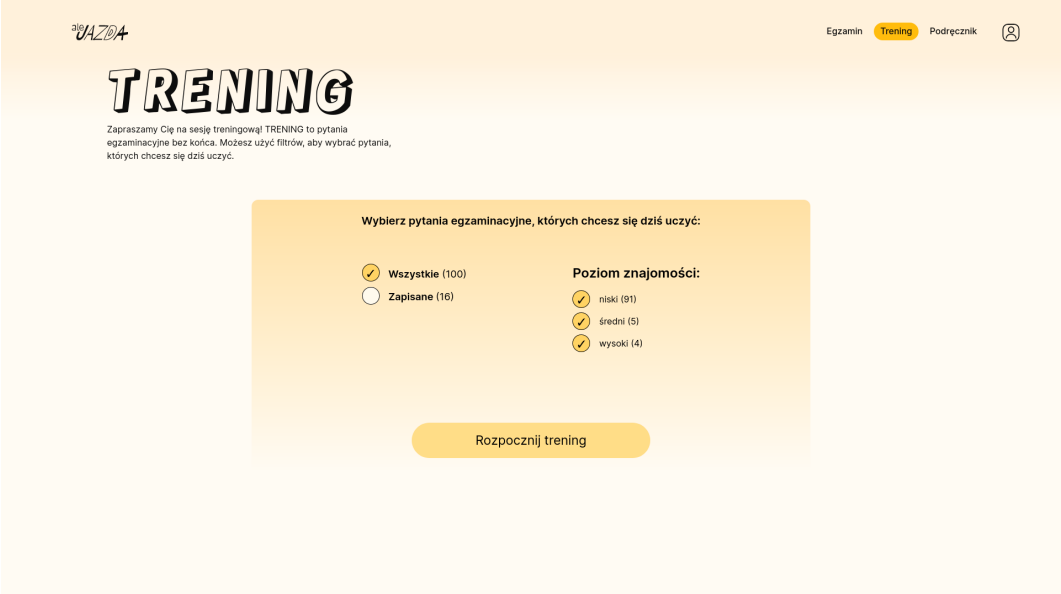


Rysunek 5.3: Wersja na komputery

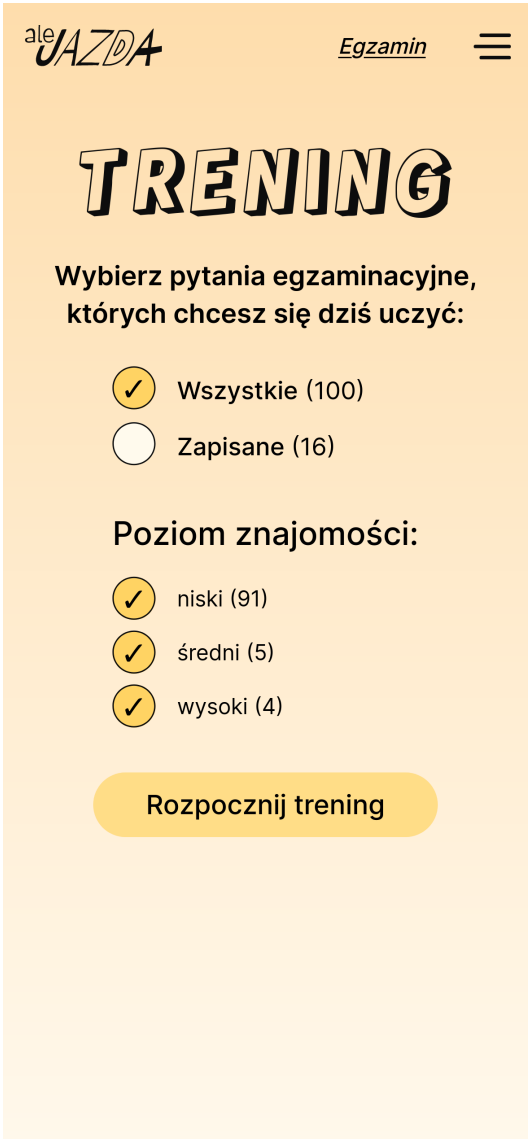
Rysunek 5.4: Wersja na urządzenia mobilne

5.2.4. Widok filtrów

Tryb treningu pozwala na losowanie pytań z ograniczonej przez użytkownika puli. Kursant może wybrać te o konkretnym poziomie znajomości spośród zapisanych lub wszystkich dostępnych pytań. Domyślnie losowane są pytania o dowolnym poziomie znajomości spośród wszystkich pytań w bazie.



Rysunek 5.5: Wersja na komputery



Rysunek 5.6: Wersja na urządzenia mobilne

5.2.5. Widok podręcznika

Podręcznik podzielono na rozdziały dotyczące wybranej tematyki, związanej z zasadami ruchu drogowego. Użytkownik może przechodzić między rozdziałami, korzystając z wysuwanego menu. Ilustracje, przedstawiające obrazy podobnej treści, np. znaki nakazu, przedstawiono w postaci listy przesuwanej na boki.

Rysunek 5.7: Wersja na komputery

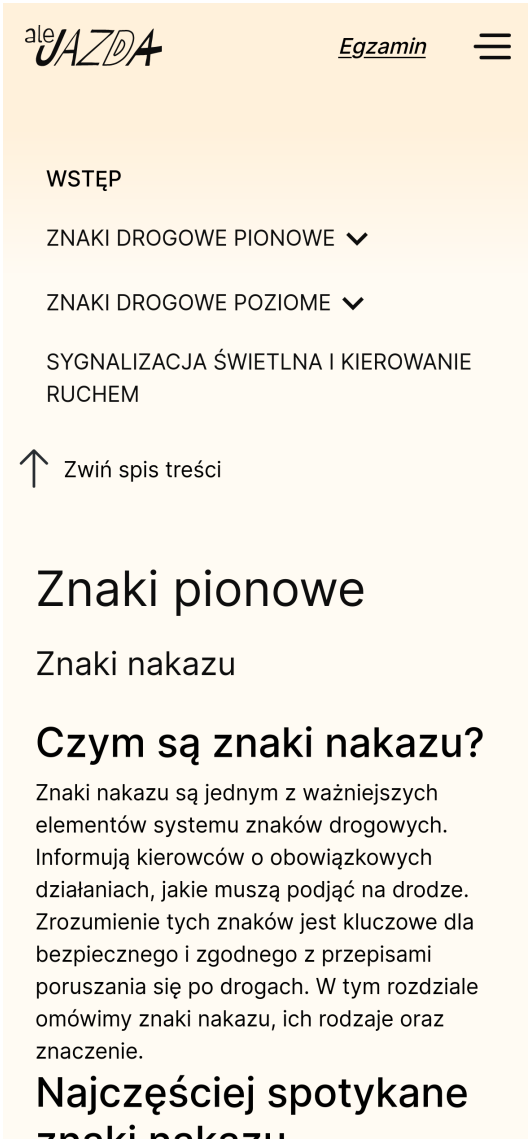
5.2.6. Widok treningu i egzaminu

Trening oraz Egzamin to widoki dzielące ten sam projekt, różniące się jedynie szczegółami. Na rysunkach 5.11 i 5.12 przedstawiono widok Treningu, na rysunkach 5.9 i 5.10 widok Egzaminu.

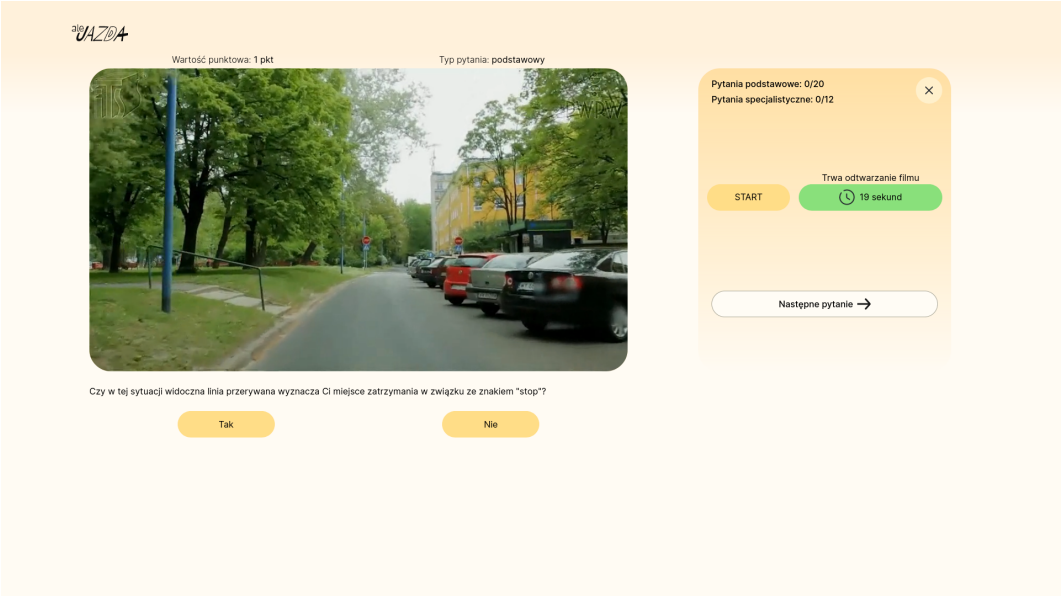
Oba widoki są podzielone na dwie sekcje: nawigacja oraz treść pytania i dołączone do niego media. Ponad pytaniem wyświetlana jest informacja o jego punktacji oraz typie pytania.

Widoki różnią się w sekcji nawigacji. Egzamin jest bardziej ograniczony i zawiera jedynie przycisk *Start*, wyświetlający media, zegar, przycisk przenoszący do następnego pytania oraz przycisk wyjścia z egzaminu. Na górze menu wyświetlany jest licznik pytań z podziałem na kategorie.

W treningu dodano pole wyboru poziomu znajomości pytania. Obok przycisku *Następne pytanie* pojawia się też odsyłacz do poprzedniego pytania. Ostatnią różnicą jest przycisk z gwiazdką, służący do zapisania pytania.

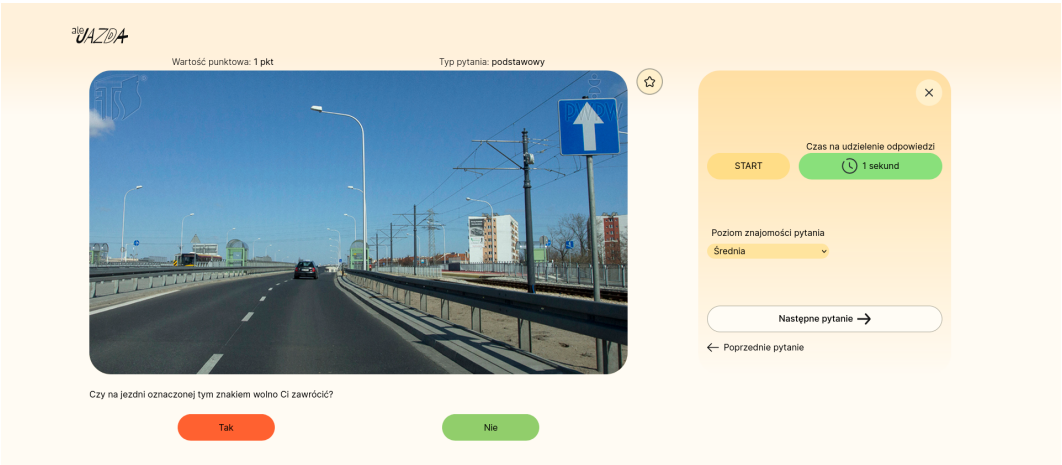


Rysunek 5.8: Wersja na urządzenia mobilne

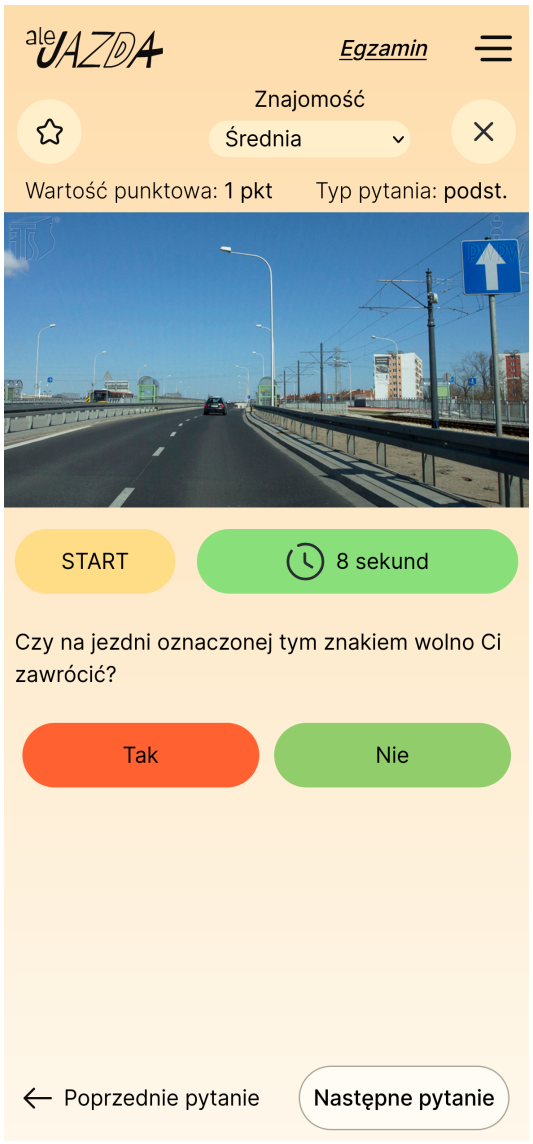


Rysunek 5.9: Wersja Egzaminu na komputery

Rysunek 5.10: Wersja Egzaminu na urządzenia mobilne



Rysunek 5.11: Wersja Treningu na komputery

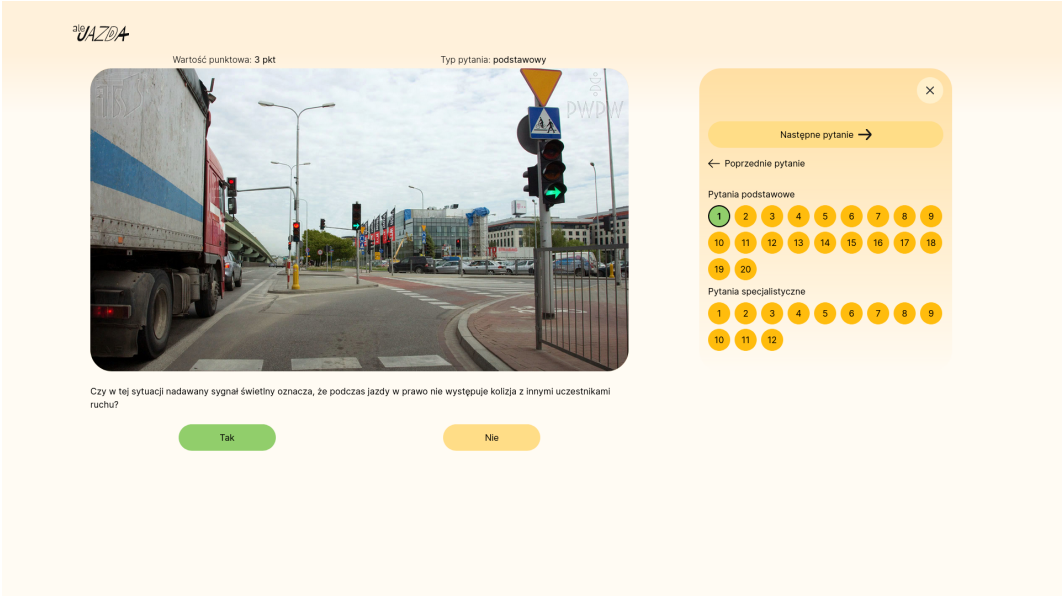


Rysunek 5.12: Wersja Treningu na urządzenia mobilne

5.2.7. Widok przeglądu egzaminu

Widok przeglądu egzaminu jest wyświetlany po zakończonym egzaminie i umożliwia ponowny przegląd pytań. Lista pytań zawiera okrągłe przyciski z numerem pytania, wypełnione odpowiednio czerwonym, zielonym i pomarańczowym kolorem, dla pytań, na które użytkownik odpowiedział niepoprawnie, prawidłowo lub nie udzielił żadnej odpowiedzi. Użytkownik wybiera interesujące go pytanie, klikając w odpowiedni element listy.

Pytania są wyświetlane podobnie jak w Egzaminie, jednak dobra i zła odpowiedź są zaznaczane na odpowiednio zielony i czerwony kolor.



Rysunek 5.13: Wersja na komputery

Rysunek 5.14: Wersja na urządzenia mobilne

5.2.8. Widok profilu użytkownika

Profil użytkownika podzielono na trzy obszary: statystyki, zapisane pytania i ustawienia. W poniższych podsekcjach zawarto krótkie opisy zawartości oraz zrzuty ekranu dla wersji na urządzenia stacjonarne i mobilne.

Statystyki

Sekcja statystyk zawiera dwa animowane wykresy, przedstawiające procent opanowanego materiału oraz liczbę egzaminów ukończonych w danym dniu. Pierwszy z nich umożliwia modyfikację wyświetlanej zawartości poprzez klikanie elementów w legendzie wykresu. Drugi pozwala na zmianę wyświetlanego tygodnia za pomocą strzałek. W widoku mobilnym naraz wyświetlany jest tylko jeden wykres. Można przełączać się między nimi, klikając strzałkę. Widok dla komputerów przedstawiono na ilustracji 5.15, dla urządzeń mobilnych na ilustracji 5.16.

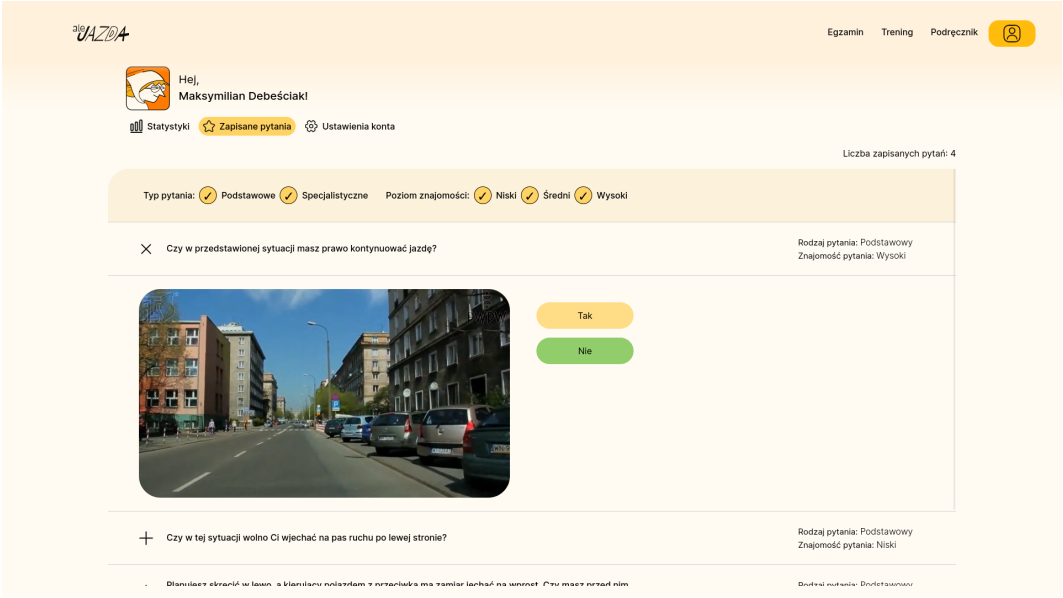
Rysunek 5.15: Wersja na komputery

Rysunek 5.16: Wersja na urządzenia mobilne

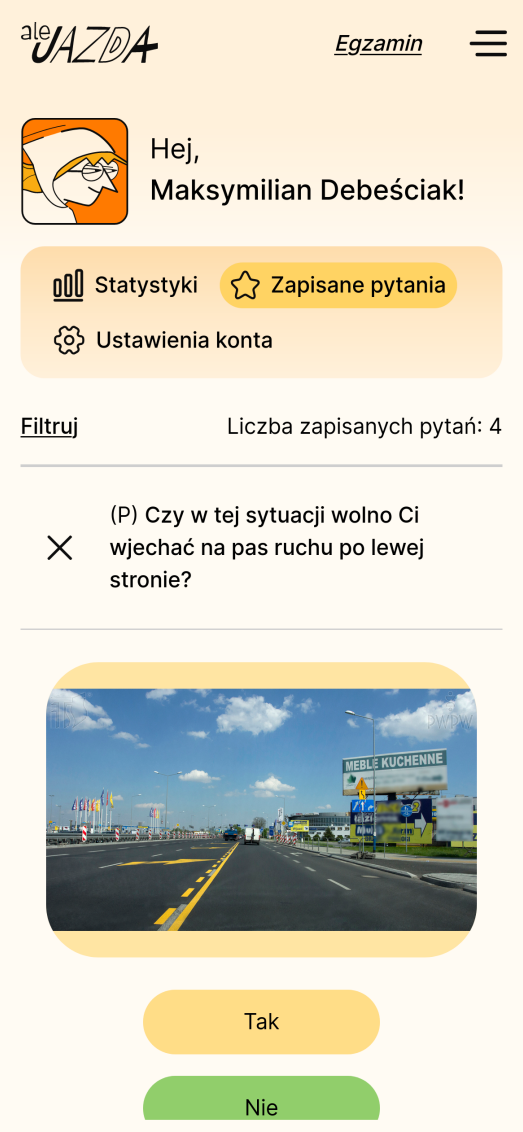
Zapisane pytania

Zawartość listy zapisanych pytań można filtrować, kierując się poziomem znajomości lub kategorią pytania. Obok znajduje się też liczba wyświetlanych pytań. W wersji mobilnej filtry są wyświetlane jako osobny panel, przykrywający pozostałą zawartość widoku.

Każdy element listy można rozwinąć, by wyświetlić media oraz odpowiedzi. W nagłówku elementu można znaleźć treść pytania, punktację i kategorię. W wersji mobilnej te informacje są ograniczone tylko do kategorii i treści pytania, a kategoria jest wyświetlana jako pojedyncza litera. Media w postaci wideo są uruchamiane przez użytkownika i umożliwiają przewijanie, zatrzymywanie, zmianę prędkości odtwarzania, przejście do trybu pełnoekranowego oraz pobranie nagrania. Widok dla komputerów przedstawiono na ilustracji 5.17, dla urządzeń mobilnych na ilustracji 5.18.



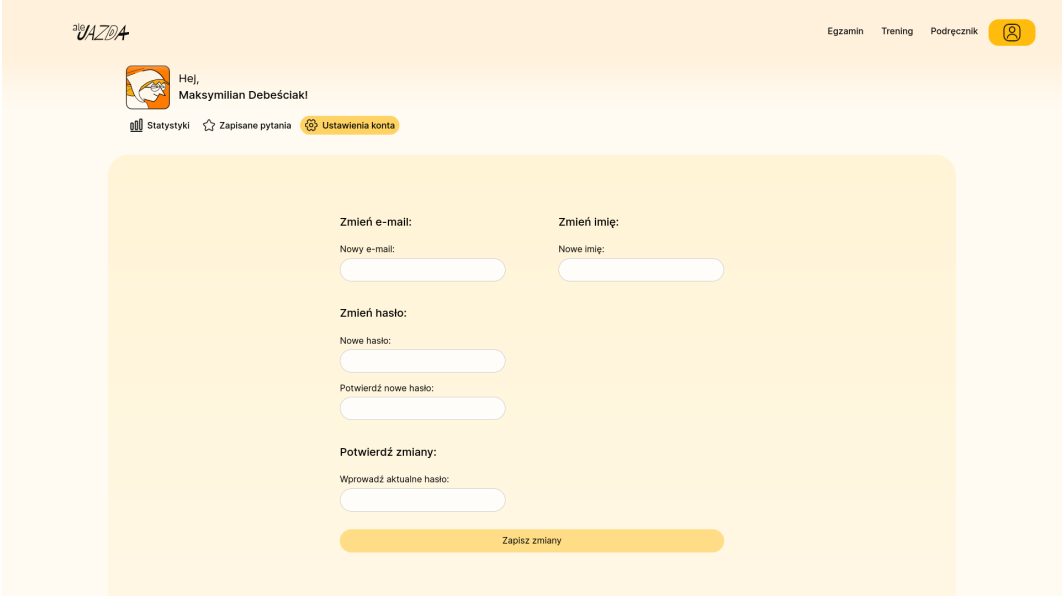
Rysunek 5.17: Wersja na komputery



Rysunek 5.18: Wersja na urządzenia mobilne

Ustawienia

Widok ustawień wyświetla pola tekstowe, w które użytkownik może wprowadzić nową nazwę użytkownika, e-mail lub hasło. Zmiany należy potwierdzić hasłem i wciśnięciem przycisku. Pola, które nie zostaną wypełnione, nie spowodują zmiany danych użytkownika. Widok dla komputerów przedstawiono na ilustracji 5.19, dla urządzeń mobilnych na ilustracji 5.20.



Rysunek 5.19: Wersja na komputery

Rysunek 5.20: Wersja na urządzenia mobilne

Rozdział 6.

Szczegóły implementacji strony serwerowej

6.1. Omówienie

Aplikacja służy użytkownikom jako pomoc w przygotowaniach do egzaminu na prawo jazdy, dlatego konieczne było przygotowanie odpowiedniej bazy danych. Zawiera ona pytania egzaminacyjne, ale także dane użytkowników, takie jak hasła, loginy, adresy e-mail, zapisane pytania, czy egzaminy. Już na wczesnym etapie prac było jasne, że zaplecze serwerowe aplikacji nie będzie szczególnie skomplikowane, ale należało zadbać o to, by jego rozbudowa w przyszłości mogła przebiegać bezproblemowo.

Kod backendu odpowiada w większości za pobieranie i aktualizowanie odpowiednich danych z bazy. Stosunkowo niewielką część stanowi rejestracja oraz uwierzytelnianie użytkowników.

6.2. Warstwowa architektura serwera

W początkowej fazie projektu kod serwera stanowił jedynie marginalną część bazy kodu, dlatego całą logikę zawarliśmy w jednym pliku. Wraz z rozbudową bazy danych oraz dodawania kolejnych funkcji do aplikacji rosła liczba zapytań do serwera i endpointów, a logika ich obsługi znacznie się komplikowała. Utrzymywanie jednego dużego pliku stało się niewygodne i nieczytelne, dlatego zaczęliśmy szukać innego rozwiązania.

W Node.js nie istnieje jeden rekomendowany sposób strukturyzacji kodu serwera. Każdy przypadek wymaga osobnego podejścia, a wiele zależy od logiki backendu, jego przeznaczenia a także preferencji zespołu. Potrzebowaliśmy rozwiązania, które zapewni nam elastyczność, przejrzystość kodu oraz łatwą rozbudowę i utrzy-

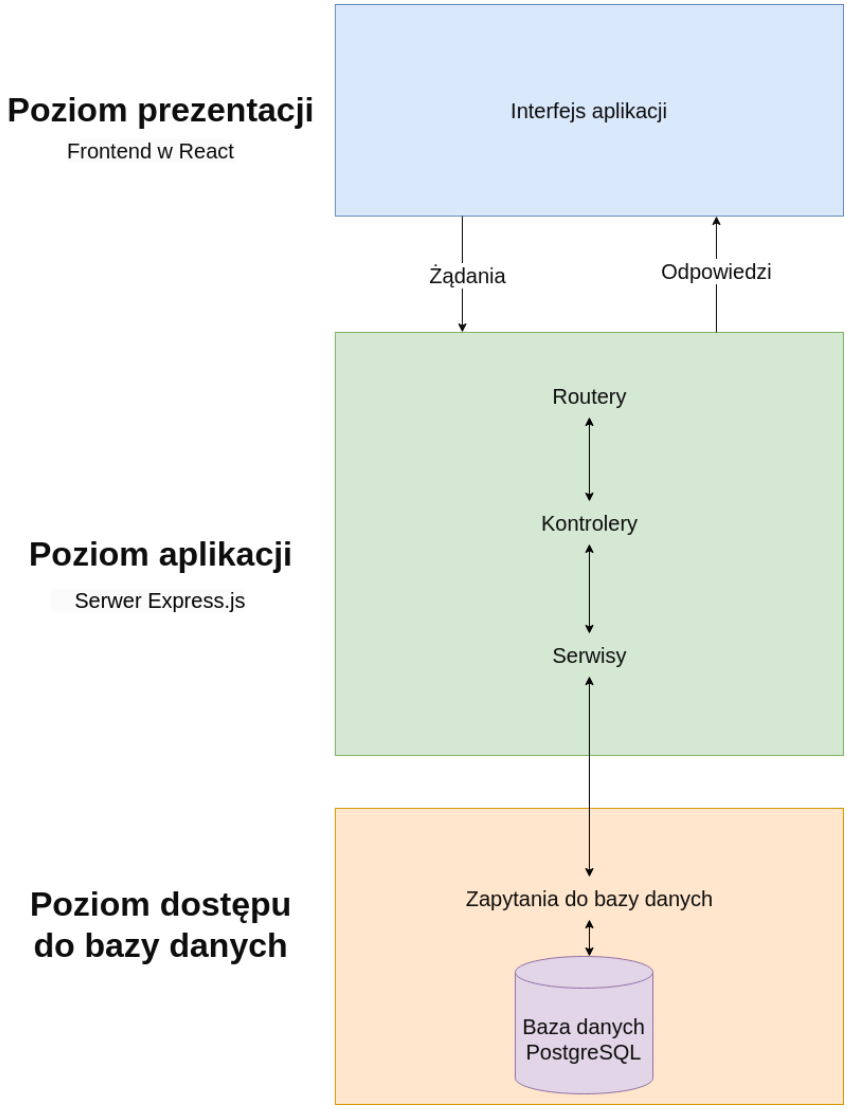
manie. Zdecydowaliśmy się znaną i dobrze udokumentowaną trójpoziomową architekturę serwera[23] (ang. *3-tier architecture*), która okazała się spełniać wszystkie nasze wymagania.

Architekturę serwera dzielimy na trzy poziomy: **prezentacji**, **aplikacji** oraz **dostepu do danych** 6.1. Każdemu z nich przypisuje się konkretną rolę. Poziom prezentacji odpowiada za wyświetlanie i zarządzanie interfejsem użytkownika, a także wysyłaniem żądań HTTP do poziomu aplikacji. Poziom aplikacji odbiera i przetwarza żądanie oraz komunikuje się z poziomem dostępu do danych, a ten jest odpowiedzialny za zarządzaniem bazą danych i pobieraniem z niej informacji. Taki podział zapewnia modularność i skalowalność, a także upraszcza zarządzanie kodem serwera. Poniżej przedstawiono przebieg komunikacji między poziomami na przykładzie mechanizmu logowania:

1. **Poziom prezentacji (React)**
   - użytkownik wprowadza dane logowania,
   - React wysyła żądanie HTTP, zawierające dane logowania, do backendu.
2. **Poziom aplikacji (Express.js)**
   - Express.js odbiera żądanie od React, i wyłuskuje z niego dane logowania
   - Express.js wysyła zapytanie o dane logowania użytkownika do poziomu dostępu do danych.
3. **Poziom dostępu do danych (PostgreSQL + Node.js**):
   - odbiera zapytanie i sprawdza, czy istnieje użytkownik o podanych danych logowania,
   - zwraca informacje do poziomu aplikacji.
4. **Poziom aplikacji (Express.js)**:
   - odbiera odpowiedź od poziomu dostępu do danych,
   - jeśli dane były prawidłowe, tworzy token autentykacji albo sesję,
   - przesyła odpowiedź do poziomu prezentacji.
5. **Poziom prezentacji (React)**:
   - React odbiera odpowiedź od poziomu aplikacji i wyświetla zawartość strony dla zalogowanego użytkownika lub informację o niepomyślnym logowaniu.

Wprowadziliśmy również podział struktury kodu serwera na kilka warstw, które dodatkowo atomizują kod i ułatwiają jego utrzymanie oraz rozwój. Struktura kodu została zainspirowana artykułem [24] i prezentuje się następująco:

- **routery** – trasują żądania HTTP do odpowiednich kontrolerów.
- **kontrolery** – wyłuskują z otrzymanego żądania dane i przekazują je do odpowiedniego serwisu.
- **serwisy** – zawierają logikę biznesową, np. odpowiadają za autoryzację użytkowników.
- **zapytania do bazy danych** – wysyłają zapytania do bazy danych, by pobrać z niej odpowiednie rekordy, dodać nowe lub zaktualizować już istniejące.

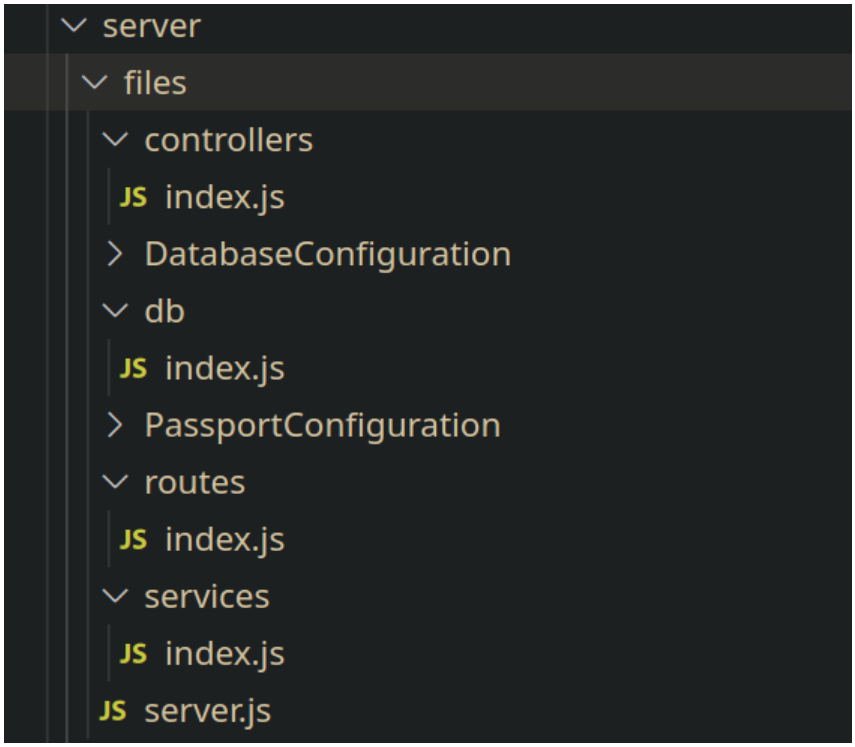


Rysunek 6.1: Trójpoziomowa architektura serwera z uwzględnieniem warstw

Takie rozwiązanie oprócz modularyzacji kodu przynosi ze sobą także inne korzyści. Zauważmy, że obsługa zapytań HTTP odbywa się jedynie na poziomie aplikacji w warstwach routerów i kontrolerów. Kolejne operują już na wyłuskanych z żądania danych. To rozdzielenie uniezależniło logikę biznesową od wyboru narzędzia do obsługi żądań. Niewielkim kosztem aktualizacji routerów oraz kontrolerów

można wymienić Express.js na inne rozwiązanie, pozostawiając pozostałe warstwy i poziomy w niezmienionym stanie.

Powstałe w trakcie przebudowy kodu pliki utworzyły czytelną i łatwą w zarządzaniu strukturę (6.2.



Rysunek 6.2: Struktura plików serwera

6.3. Baza danych

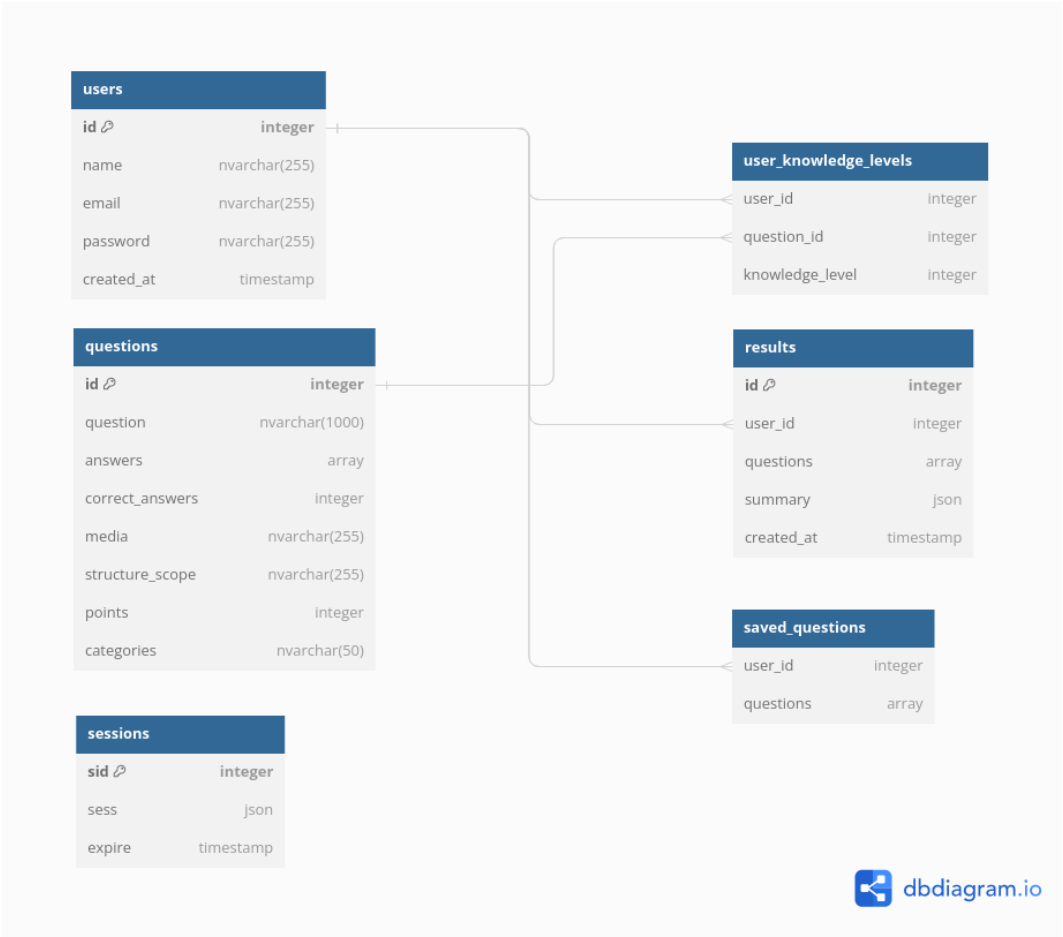
Wybraliśmy relacyjną bazę danych w PostgreSQL. Uznaliśmy to rozwiązanie za najprostsze i uniwersalne, idealne dla naszych zastosowań. Bazę danych pytań oraz użytkowników przechowujemy w chmurze w serwisie Neon [25]. Schemat bazy danych przedstawiono na rysunku 6.3.

W bazie w tabeli *questions* przechowujemy pytania egzaminacyjne, udostępnione przez Ministerstwo Infrastruktury[26]. Część z nich zawiera odnośniki do mediów (nagrania wideo lub zdjęcia), które ze względu na ograniczoną pojemność naszej bazy, zdecydowaliśmy się przechowywać w osobnym serwisie. Wybraliśmy w tym celu Amazon S3.

Dane użytkowników, tj. nazwy, hasła, adresy mailowe przechowujemy w tabeli users. Zapisane przez danego użytkownika pytania znajdują się w tabeli saved_questions, z kolei poziomy znajomości danego pytania zapisane przez danego

użytkownika w tabeli user_knowledge_levels. W tych tabelach identyfikatory użytkownika oraz pytania z tabel users i questions służą jako klucze obce.

W bazie zawarta jest także tabela *results*, która przechowuje listę pytań oraz podsumowanie egzaminu zakończonego przez danego użytkownika.



Rysunek 6.3: Schemat bazy danych

Rozdział 7.

Podsumowanie i przyszłość projektu

Realizowany projekt okazał się ambitnym przedsięwzięciem, które wymagało zaangażowania oraz zwiększenia swoich kompetencji w pracy nad częścią serwerową aplikacji, oraz jej interfejsem, projektowania interfejsu i doświadczenia użytkownika, a także zarządzania bazami danych i bazą kodu źródłowego. Obecnie aplikacji to około ośmiu tysięcy linii, w tym siedem tysięcy we frontendzie i tysiąc po stronie serwerowej 7.1. Statystyki kodu uzyskano przy pomocy narzędzia cloc[27].

Language	files	blank	comment	code
JavaScript	149	947	96	8655
CSS	3	880	165	3082
Text	3	65	0	233
HTML	3	2	0	74
SUM:	158	1894	261	12044

Rysunek 7.1: Statystyki kodu

Mimo niemałych problemów, jakie napotkaliśmy w trakcie prac, udało nam się zrealizować wszystkie podstawowe zamierzenia projektu, a dodatkowo przygotować osobną wersję interfejsu na urządzenia mobilne. Z początku planowaliśmy jedynie wprowadzić responsywność do widoku na komputery.

Jesteśmy bardzo zadowoleni z efektów naszych działań i liczymy, że zostaną one przyjęte z entuzjazmem przez przyszłych użytkowników. Zaangażowaliśmy się w ten projekt w ogromnym stopniu i nie wyobrażamy sobie, by pozostawić go w obecnym stanie. Planujemy w najbliższym czasie zaimplementować wszystkie pozostałe funkcje serwisu, przede wszystkim ukończyć podręcznik, wprowadzić większą gamę statystyk i trybów nauki. W dalszej perspektywie mamy m.in. wprowadzenie elementów grywalizacji, budowę modułu społecznościowego (komunikacja z zaprzy-

jaźnionymi użytkownikami).

W miarę rozwoju aplikacji z pewnością pojawi się konieczność zaproszenia do zespołu nowych grafików i programistów. Przeprowadziliśmy wstępne rozeznanie i zebraliśmy już grupę osób zainteresowanych współpracą.

Bibliografia

[1] Git, https://git-scm.com/, (data dostępu: 21.08.2023).

[2] Github, https://github.com/, (data dostępu: 21.08.2023).

[3] Figma, https://www.figma.com/, (data dostępu: 21.08.2023).

[4] HackMD, https://hackmd.io/, (data dostępu: 21.08.2023).

[5] Duolingo, https://www.duolingo.com/, (data dostępu: 31.08.2023).

[6] Strona Nielsen Norman Group, https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/, (data dostępu: 21.08.2023).

[7] Amazon S3, https://aws.amazon.com/s3/, (data dostępu: 22.08.2023).

[8] Amazon EC2, https://aws.amazon.com/ec2/, (data dostępu: 22.08.2023).

[9] React, https://react.dev/, (data dostępu: 21.08.2023).

[10] Tailwind CSS, https://tailwindcss.com/, (data dostępu: 20.08.2023).

[11] Tailwind Styled Components, https://github.com/MathiasGilson/tailwind-styled-component, (data dostępu: 21.08.2023).

[12] React Router, https://reactrouter.com/en/main, (data dostępu: 21.08.2023).

[13] React Redux, https://react-redux.js.org/, (data dostępu: 21.08.2023).

[14] React Context, https://legacy.reactjs.org/docs/context.html, (data dostępu: 21.08.2023).

[15] Atomowa architektura komponentów, https://github.com/danilowoz/react-atomic-design, (data dostępu: 28.08.2023).

[16] CSS Modules, https://github.com/css-modules/css-modules, (data dostępu: 27.08.2023).

[17] Styled Components, https://styled-components.com/, (data dostępu: 27.08.2023).

[18] Progressive web apps, https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps, (data dostępu: 30.08.2023).

[19] Node.js, https://nodejs.org/en, (data dostępu: 27.08.2023).

[20] Express.js, https://expressjs.com/, (data dostępu: 27.08.2023).

[21] PostgreSQL, https://www.postgresql.org/, (data dostępu: 27.08.2023).

[22] Passport.js, https://www.passportjs.org/, (data dostępu: 27.08.2023).

[23] Trójpoziomowa architektura serwera, https://www.tonymarston.net/php-mysql/3-tier-architecture.html, (data dostępu: 27.08.2023).

[24] Organizacja kodu serwera, https://www.coreycleary.me/project-structure-for-an-express-rest-api-when-there-is-no-standard-way/, (data dostępu: 27.08.2023).

[25] Neon, https://neon.tech, (data dostępu: 21.08.2023).

[26] Baza pytań egzaminacyjnych, https://www.gov.pl/web/infrastruktura/prawo-jazdy, (data dostępu: 27.08.2023).

[27] Narzędzie do wyznaczania statystyk kodu, https://github.com/AlDanial/cloc, (data dostępu: 28.08.2023).