Uniwersytet Łódzki

Wydział Matematyki i Informatyki

Kierunek Analiza danych

Praca dyplomowa inżynierska

Projektowanie aplikacji internetowych z wykorzystaniem frameworka Django

Autor: inż. Damian Wąsik

Promotor pracy: dr Piotr Fulmański

Łódź, 2021

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Szybki postęp technologiczny oznacza, że możliwości realizacji pomysłów przy użyciu powszechnych dziś narzędzi są nieograniczone. Mając określoną ilość czasu i pieniędzy, ludzie mogą używać aplikacji do rozwiązywania wszelkiego rodzaju problemów. Z aplikacji korzystamy na co dzień. Dzięki odpowiedniemu doborowi i sprytnemu wykorzystaniu, mogą nam bardzo wygodnie zaoszczędzić sporo czasu (a czas jest najcenniejszym zasobem). Projektowanie aplikacji jest dla mnie fascynującym przeżyciem i postaram się przekazać w swojej pracy jak najwięcej wartościowych informacji na ten temat.

1.2 Cel pracy

W mojej pracy zawarte są informacje na temat projektowania aplikacji internetowej. Przed rozpoczęciem pisania założyłem, że opiszę krok po kroku jak wygląda proces tworzenia rozwiązania danego problemu, opiszę narzędzia jakie są potrzebne oraz całe środowisko pracy. Aby osoba, która nie specjalizowała się w tym temacie, po przeczytaniu tego tekstu była w stanie zainteresować się tematem i podjąć pierwsze kroki w oparciu o zawartą w nim wiedzę. Wszystko na przykładzie mojego projektu aplikacji webowej dzięki czemu można od razu zwizualizować prezentowaną wiedzę.

1.3 Struktura pracy

Rozdział 2

Aplikacje internetowe

2.1 Rodzaje aplikacji

Najbardziej popularne kategorie aplikacji [2] to:

1. aplikacje internetowe
2. aplikacje desktopowe
3. aplikacje mobilne
4. aplikacje wiersza poleceń

2.1.1 Aplikacje internetowe

Aplikacja internetowa to program komputerowy, który wykorzystuje przeglądarkę internetową do wykonywania pewnych funkcji. Nazywana jest również aplikacją internetową. Aplikacje webowe są obecne na wielu stronach internetowych.

Jak działają aplikacje internetowe?

Aplikacja webowa jest programem typu klient-serwer. Oznacza to, że ma stronę klienta i stronę serwera. Termin "klient" odnosi się tutaj do programu, którego dana osoba używa do uruchomienia aplikacji. Jest on częścią środowiska klient-serwer, w którym wiele komputerów współdzieli informacje. Na przykład, w przypadku bazy danych, klient jest programem, za pomocą którego użytkownik wprowadza dane. Serwer jest

aplikacją, która przechowuje te informacje. Są one dostępne z poziomu przeglądarki internetowej (takiej jak Google Chrome, Safari, czy Firefox). Przeglądarki są wbudowane w każdy popularny system operacyjny. Ich ogromną zaletą z perspektywy użytkownika jest wysoka dostępność. Żeby skorzystać aplikacji webowej, wystarczy znać jej adres internetowy. Przykładem aplikacji internetowej jest YouTube [1].

Wykorzystanie aplikacji internetowych w życiu codziennym

Przedsiębiorstwa muszą wymieniać informacje i zawierać transakcje ze swoimi docelowymi klientami. Internet może być doskonałym i niedrogim kanałem do tego celu, pod warunkiem, że istnieje sposób na przechwytywanie i przechowywanie wszystkich niezbędnych danych oraz pokazywanie wyników użytkownikom. Dzięki aplikacjom internetowym, użytkownicy mogą wchodzić w interakcję z firmą za pomocą koszyków na zakupy lub systemów zarządzania treścią. Aplikacje internetowe mogą być tworzone z wielu różnych powodów i wykorzystywane przez firmy lub osoby prywatne. Osoby prywatne potrzebują ich, aby ułatwić sobie komunikację lub kupować rzeczy online. Również pracownicy mogą współpracować nad projektami i pracować na wspólnych dokumentach z aplikacjami internetowymi. Mogą tworzyć raporty, pliki i udostępniać informacje z dowolnego miejsca i urządzenia.

2.1.2 Aplikacje desktopowe

Jest to oprogramowanie instalowane bezpośrednio na komputerze osobistym. Można je uruchomić w dowolnym momencie, niezależnie od innych aplikacji. Nie zajmują miejsca na dysku i mogą działać niezależnie od połączenia internetowego. Mimo że niektóre aplikacje potrzebują go do działania zgodnie z przeznaczeniem (na przykład przeglądarki internetowe, takie jak Chrome lub Firefox), są one nadal uważane za aplikacje desktopowe, ponieważ są instalowane na komputerze.

Są one dystrybuowane na każdy system operacyjny oddzielnie (np. program napisany na system Windows nie zadziała na Linux). Mogą być szybsze w działaniu, ponieważ zazwyczaj nie są uzależnione od szybkości łącza internetowego. Poza wspomnianymi wcześniej przeglądarkami internetowymi istnieje mnóstwo innych aplikacji desktopowych, które są dobrze znane. Programy biurowe, takie jak Word, Excel i Powerpoint, edytory graficzne, takie jak Photoshop i Paint, gry wideo, odtwarzacze multimedialne. Niektóre z nich mają więcej funkcji, niektóre są bardziej zasobożerne, a niektóre są dość proste. Ale jedną wspólną cechą wszystkich jest to, że są zainstalowane na pulpicie i

zapewniają jakąś funkcjonalność.

2.1.3 Aplikacje mobilne

Aplikacje mobilne odchodzą od zintegrowanych systemów oprogramowania, jakie zazwyczaj można znaleźć na komputerach stacjonarnych lub na stronach internetowych. Zamiast tego, każda aplikacja zapewnia ograniczoną i wyizolowaną funkcjonalność, taką jak gra, kalkulator lub mobilne przeglądanie stron internetowych.

We wdrażaniu aplikacji mobilnych wykorzystuje się dwa podejścia:

1. wyodrębnienie części funkcji aplikacji podstawowej, które są następnie zaimplementowane w aplikacji moblilnej

2. postawienie na tworzenie od zera aplikacji pod potrzeby użytkownika mobilnego (tzw. mobile first)

1. Najprostsze aplikacje mobilne wykorzystują funkcje aplikacji desktopowych lub webowych i przenoszą je na urządzenia mobilne. Stanowi to naturalne rozszerzenie grupy docelowej klientów na tych, którym wygodniej korzystać z systemu na urządzeniu mobilnym. W miarę jak aplikacje mobilne stają się coraz bardziej rozbudowane, technika ta traci na znaczeniu.

2. Bardziej wyrafinowane podejście polega na tworzeniu aplikacji specjalnie dla środowiska mobilnego, wykorzystując zarówno jego ograniczenia, jak i zalety. Na przykład, aplikacje wykorzystujące funkcje lokalizacyjne są z natury rzeczy tworzone od podstaw z myślą o urządzeniach mobilnych, ponieważ użytkownik nie jest przywiązany do lokalizacji, jak w przypadku komputerów stacjonarnych. Przykładem takiego rozwiązania może być aplikacja Nozbe, do zarządzania projektami i zadaniami. Użytkownicy tego produktu realizują zadania codziennie niezależnie od tego czy akurat siedzą w pracy przed komputerem, czy są na zakupach i potrzebują listy, którą wygodnie im przechowywać na smartfonie.

Aplikacje mobilne są używane codziennie przez użytkowników smartfonów. Dzięki nim we własnej kieszeni mamy dostęp do wszystkich potrzebnych nam szybko informacji.

2.1.4 Aplikacje wiersza poleceń

Aplikacje wiersza poleceń to programy bez interfejsu graficznego. To metoda interakcji z aplikacjami i systemami operacyjnymi, używana do wykonywania określonych zadań wymaganych przez użytkowników.

CLI [1] jest interfejsem tekstowym, w odróżnieniu od GUI [2], który wykorzystuje opcje graficzne, które umożliwiają użytkownikowi interakcję z systemem operacyjnym i aplikacjami. Program uruchamia się komendą uruchomieniową (zazwyczaj też jest opcja podania argumentów uruchomieniowych w celu modyfikacji zachowania aplikacji).

Jego mechanizm działania jest bardzo prosty, ale nie jest przyjazny dla użytkownika. Użytkownik wpisuje określone polecenie, naciska "Enter", a następnie czeka na odpowiedź. Po otrzymaniu polecenia, CLI przetwarza je odpowiednio i pokazuje wynik na tym samym ekranie. Programy linii poleceń uruchamia się w powłoce [3]

Aby jak najlepiej wykorzystać CLI, użytkownik musi być w stanie szybko wprowadzać wiązkę poleceń (jedno po drugim). Dlatego zazwyczaj są wykorzystywane przez programistów, administratorów. Są oni przyzwyczajeni do tego rodzaju programów, ponieważ potrafią szybko posługiwać się komendami i ułatwiają im pracę.

Tego typu aplikacje są najszybszym sposobem na dostarczenie rozwiązania problemu. Nie potrzeba projektu graficznego, rozkładu elementów, wystarczy tylko sam tekst.

2.2 W jaki sposób działa aplikacja internetowa?

Budując aplikację internetową, należy pamiętać o trzech głównych zasadach. Z punktu widzenia klienta, aplikacja powinna być prosta, estetyczna i rozwiązywać większość jego problemów. Z punktu widzenia biznesu, aplikacja internetowa powinna być dopasowana do swojego produktu/rynku. Z perspektywy inżynierii oprogramowania, aplikacja internetowa powinna być skalowalna, funkcjonalna i wytrzymać duże obciążenie ruchem.

[1] (ang. command line interface) - interfejs wiersza poleceń

[2] (ang. graphical user interface) - graficzny interfejs użytkownika

[3] (ang. shell) – program komputerowy pełniący rolę pośrednika pomiędzy systemem operacyjnym lub aplikacjami a użytkownikiem, przyjmując jego polecenia i „wyprowadzając" wyniki działania programów

2.2.1 Podział na front-end i back-end

Programowanie aplikacji internetowych dzieli się na dwie główne dziedziny:

- front-end
- back-end

Front-end i back-end to terminy, które pojawiają się podczas tworzenia aplikacji internetowych. Terminy te odnoszą się do elementów architektonicznych i warstw, które składają się na te aplikacje. Zazwyczaj za każdą dziedzinę jest odpowiedzialny oddzielny zespół programistów. Ułatwia to w ogólnym zarządzaniu w projekcie.

Co to jest front-end?

Front-end odpowiada za wszystko, co użytkownik widzi jak się czuje korzystając z aplikacji. Za to, jak strona wygląda, jakie są na niej animacje, przejścia. Jak elementy są ułożone na stronie, jaka jest szata kolorystyczna. Programiści front-end piszą kod źródłowy serwisu w języku HTML, stylizują wygląd strony w CSS oraz odpowiadają za doświadczenie użytkownika na stronie za pomocą języka JavaScript.

Backend jest powiązany z całą logiką aplikacji. Wartości widoczne na stronie muszą być odpowiednio przetworzone i wyekstrahowane np. z bazy danych. Zespół back-end odpowiada za następujące zadania:

- komunikacja z bazami danych
- bezpieczeństwo danych
- tworzenie interfejsów API

2.2.2 Architektura client-server

Aplikacje internetowe są dostępne za pomocą protokołu HTTP[4]. Jest on zaprojektowany na bazie architektury client-server. Model client-server jest rozproszoną strukturą aplikacji, która dzieli zadania lub obciążenie pracą pomiędzy dostawców zasobów lub usług, zwanych serwerami, a żądających usług, zwanych klientami. W architekturze

[4]Hypertext Transfer Protocol

client-server, gdy komputer kliencki wysyła żądanie danych do serwera przez Internet, serwer akceptuje żądany proces i dostarcza żądane pakiety danych z powrotem do klienta. Najprostszym zapytaniem jest po prostu adres internetowy strony.

Działająca architektura client-server umożliwia następujący proces:

1. Klient przesyła żądanie za pośrednictwem urządzenia podłączonego do sieci.
2. Serwer sieciowy odbiera i przetwarza żądanie.
3. Serwer dostarcza odpowiedź do klienta.

2.2.3 API

API[5] jest to interfejs umożliwiający wymianę danych pomiędzy aplikacjami. Zacznijmy od prostego przykładu: komunikacji międzyludzkiej. Możemy wyrażać nasze myśli, potrzeby i pomysły za pomocą języka (pisanego i mówionego), gestów lub mimiki twarzy. Interakcja człowieka z komputerami, aplikacjami i stronami internetowymi wymaga zazwyczaj elementów interfejsu użytkownika - ekranu z menu i elementami graficznymi, klawiatury i myszy. Oprogramowanie lub jego elementy nie potrzebują graficznego interfejsu użytkownika, aby komunikować się ze sobą. Produkty z dziedziny oprogramowania wymieniają się danymi i funkcjonalnościami za pomocą interfejsów odczytywanych maszynowo - API (application programming interfaces).

Może to być ustandaryzowany sposób komunikacji pomiędzy programistami. Sposobem na wykorzystanie API może być komunikacja pomiędzy zespołem front-end i back-end. Zespoły backend przetwarzają dane i wystawiają tzw. końcówki HTTP (ang. endpoints) z zestawem danych gotowych do wyświetlenia na stronę kliencką.

Przykłady API, z których korzystamy na co dzień:

1. dane pogodowe
2. logowanie za pośrednictwem serwisów społecznościowych

1. Jednym z powszechnych przykładów użycia API, z którym spotykamy się na co dzień, są dane pogodowe. Bogate widżety pogodowe wydają się być powszechne, można je znaleźć na wszystkich platformach, takich jak Google Search, Apple Weather, a

[5](ang. application programming interface) - interfejs programowania aplikacji

nawet z Twojego inteligentnego urządzenia domowego. Na przykład, jeśli wyszukujesz "pogoda + [nazwa Twojego miasta]" w Google, zobaczysz dedykowane pole na górze wyników wyszukiwania z aktualnymi warunkami pogodowymi i prognozą.

2. Innym znanym przykładem użycia API jest funkcja "zaloguj się używając Facebooka/Twittera/Google'a/Githuba", którą można zobaczyć na wielu stronach internetowych. Zamiast faktycznego logowania się do kont użytkowników w mediach społecznościowych (co stanowiłoby poważny problem bezpieczeństwa), aplikacje z tą funkcjonalnością wykorzystują API tych platform do uwierzytelniania użytkownika przy każdym logowaniu. Sposób w jaki to działa jest dość prosty. Za każdym razem, gdy aplikacja się ładuje, używa API do sprawdzenia, czy użytkownik jest już zalogowany za pomocą jakiejkolwiek platformy mediów społecznościowych. Jeśli nie, gdy użytkownik kliknie przycisk np. "Zaloguj się za pomocą Facebook", otwiera się okienko, w którym jest proszony o potwierdzenie, że rzeczywiście chce się zalogować za pomocą tego profilu mediów społecznościowych. Gdy użytkownik potwierdzi, API dostarcza aplikacji informacje identyfikacyjne, dzięki czemu wie, kto się loguje.

2.3 Frameworki

W rozwoju aplikacji internetowych prócz znajomości języków programowania wykorzystuje się różne frameworki. Framework to przygotowany zestaw narzędzi zaprojektowany po to, żeby ułatwiać rozwiązanie danego problemu. Te szkielety aplikacji internetowych oferują szeroką gamę gotowych komponentów, fragmentów kodu i całych szablonów aplikacji. Frameworki mogą być używane do tworzenia usług internetowych, API i innych zasobów internetowych. Popularne frameworki są utrzymywane przez zespół programistów i często posiadają obszerną dokumentację z przykładami użycia. Zostały również dobrze przetestowane. Z tego powodu wszystko związane z utrzymaniem tego kodu jest po stronie innego zespołu. Oszczędza też dużo czasu. Nie ma potrezeby zastanawiać się nad przygotowaniem niezbędnych narzędzi, ale bezpośrednio tworzyć logikę aplikacji.

2.3.1 Charakterystyka frameworków

Zalety korzystania z frameworków:

- sprawiają, że proces rozwoju i utrzymania jest znacznie szybszy i łatwiejszy.

- zawierają ustandaryzowane praktyki kodowania i konwencje dla struktur kodu.
- pozwalają usystematyzować proces programowania programistów i uniknąć błędów i błędów.
- pozwalają skupić się całkowicie na swojej aplikacji, dbając o wszystkie szczegóły, takie jak zarządzanie danymi i konfiguracją skutecznie.

Podobnie jak w przypadku podziału architektury na dwa obszary, czyli front-end i back-end, frameworki posiadają również podział:

1. frameworki client-side
2. frameworki server-side

Podczas gdy frameworki po stronie klienta są używane do obsługi interfejsu użytkownika, frameworki po stronie serwera działają w tle, aby zapewnić płynne funkcjonowanie strony internetowej.

Podział frameworków ze względu na najpopularniejsze wzorce architektoniczne:

- Model View Controller
- Model-View-ViewModel (MVVM)
- Push-based vs Pull-based
- Three-tier organization

Przykłady najpopularniejszych frameworków

Frameworki client-side:

- React
- Angular
- Vue.js
- jQuery

Frameworki server-side:

- Django (Python)
- Flask (Python)
- Express.js (JavaScript)
- Spring (JavaScript)
- Laravel (PHP)
- CakePHP (PHP)
- Ruby on Rails (Ruby)

2.3.2 Framework Django

Django to wysokopoziomowy framework napisany w języku Python, który umożliwia szybkie tworzenie bezpiecznych i łatwych w utrzymaniu stron internetowych. Zbudowany przez doświadczonych programistów, Django zajmuje się wieloma kłopotami związanymi z tworzeniem stron internetowych, więc można skupić się na pisaniu swojej aplikacji bez potrzeby wymyślania koła na nowo. Jest darmowy i otwarty, ma prężną i aktywną społeczność, świetną dokumentację i wiele opcji darmowego i płatnego wsparcia.

2.4 Prototyp aplikacji

Prototyp aplikacji to schematyczny projekt interfejsu użytkownika. Składa się ze szkiców ekranowych systemu projektowego. Forma nie ma znaczenia - można nawet narysować ją ręcznie na papierze lub skorzystać ze specjalistycznych narzędzi projektowych. Prototypowanie to świetny sposób na szybkie projektowanie aplikacji internetowych i mobilnych.

2.4.1 Tworzenie prototypu

W którym momencie procesu wytwarzania oprogramowania należy tworzyć prototypy?

Jest to jeden z kluczowych elementów w fazie projektowania aplikacji webowych i mobilnych. Aby rozpocząć proces prototypowania, należy najpierw zdefiniować wymagania, następnie je przeanalizować, a następnie przystąpić do szkicowania ekranów aplikacji lub strony internetowej. Najważniejszy moment nadchodzi zaraz potem. Otóż, najważniejszą rzeczą przy tworzeniu prototypu jest zweryfikowanie go z użytkownikiem systemu. Bez tego elementu praca tworzenia makiet jest bezcelowa. Informacje zwrotne zebrane w fazie prototypowania prowadzą z powrotem do analizy i definicji wymagań. Dzieje się tak, ponieważ można na wczesnym etapie zidentyfikować luki funkcjonalne w powstałym produkcie, szybko je uzupełnić, a następnie ponownie zainstalować w prototypie.

Dlaczego warto tworzyć prototyp aplikacji przed rozpoczęciem prac nad nią?

Porównajmy systemy informacji budowlanej używane do budowy domów. Na etapie papieru i ołówka łatwo zamienić kuchnię w sypialnię, a łazienkę w garaż. Jednak po wybudowaniu domu wszystkim znacznie trudniej jest wprowadzić te zmiany. Poza tym tworzenie prototypów jest bardzo tanie, a stworzenie zarysu działania systemu nie zajmuje dużo czasu. Wystarczy kilka szkiców i screenów, aby pokazać wizję potencjalnym użytkownikom. Po zebraniu i przeanalizowaniu informacji zwrotnej, można skupić się na poprawie niektórych elementów aplikacji, a następnie rozpocząć pracę wraz z gotowym schematem działania.

2.4.2 Badanie rynku

Badania rynku definiuje się jako proces oceny wykonalności nowego produktu lub usługi, poprzez badania prowadzone bezpośrednio z potencjalnymi konsumentami. Metoda ta pozwala organizacjom lub firmom odkryć ich rynek docelowy, zebrać i udokumentować opinie oraz podjąć świadome decyzje. W momencie posiadania prototypu rozwiązania wymyślonego problemu, warto zastanowić się nad podjęciem działań w kierunku poznania rynku i swojej grupy odbiorców. Znajomość grupy docelowej jest pierwszym krokiem do sprawdzenia, czy pomysł na produkt jest idealnym przedsięwzięciem, czy nie. To właśnie tutaj można by określić, czy istnieje rzeczywisty rynek z konsumentami, którzy potrzebują i będą płacić za produkt.

2.4.3 MVP

MVP[6] to wersja nowego produktu, która pozwala zespołowi zebrać maksymalną ilość zweryfikowanej wiedzy o kliencie przy najmniejszym nakładzie pracy. Jest to rozwiązanie, które nie jest kompletne, ale zawiera wystarczająco dużo funkcji, aby zaspokoić podstawową potrzebę klienta i umożliwić testowanie kolejnych funkcji z wykorzystaniem informacji zwrotnych pochodzących bezpośrednio z rynku.

2.5 Architektura aplikacji

Architektura aplikacji to projekt organizacyjny całego systemu oprogramowania, w tym wszystkich jego komponentów i elementów zewnętrznych. Istnieją różne wzorce projektowe, które są używane do definiowania tego typu architektury, a wzorce te pomagają przekazać, w jaki sposób aplikacja będzie realizować niezbędne procesy biznesowe zdefiniowane w wymaganiach systemowych.

2.5.1 Programowanie obiektowe

Programowanie zorientowane obiektowo (OOP) to paradygmat programowania, który opiera się na koncepcji klas i obiektów. Służy on do podziału programu na proste, nadające się do wielokrotnego użytku fragmenty kodu (zwykle nazywane klasami), które są używane do tworzenia poszczególnych instancji obiektów. Przy tworzeniu aplikacji internetowych jest to najpopularniejsze podejście organizowania kodu.

2.5.2 Organizacja katalogów

Jeśli chodzi o maszynę, organizacja jest w większości przypadków nieistotna. Z perspektywy komputera nie ma znaczenia, czy umieścisz cały swój kod w jednej metodzie składającej się z miliona linijek, czy klasy będą posortowane alfabetycznie, czy wszystkie zmienne będą miały jednoliterowe nazwy. W organizowaniu kodu nie chodzi o komunikację z komputerem, ale o to, by ludzie rozumieli kod na tyle dobrze, by mogli go utrzymywać i rozwijać z pewnym stopniem wydajności i komfortu.

[6](ang. minimum viable product)

Gdy jednostka kodu rozrasta się i zawiera zbyt wiele elementów, nawigacja, przeglądanie i zrozumienie stają się trudne. Sposobem na rozwiązanie tego problemu jest rozbicie jednostki na mniejsze, łatwe w zarządzaniu części, które są logicznie ze sobą powiązane. W przypadku klas dość dobrze wiadomo, że należy to zrobić w taki sposób, aby tworzyły obiekty logiczne, które mają dobrą spójność i dobrze pasują do modelu domeny.

2.5.3 Dobór narzędzi i technologii

2.5.4 Zarządzanie zależnościami

2.5.5 Aplikacje typu "monolit"

2.5.6 Aplikacje rozproszone

2.6 Wdrażanie i uruchamianie aplikacji

2.6.1 Uruchomienie aplikacji w chmurze

2.6.2 Testowanie aplikacji

2.6.3 Monitoring procesów

2.6.4 Bezpieczeństwo

2.6.5 Skalowanie aplikacji

Rozdział 3

Zarządzanie danymi w aplikacji

3.1 Sposoby pozyskiwania danych

3.1.1 Web scraping

Web scraping to technika pozwalająca na pozyskiwanie ogólnodostępnych w sieci danych. Dzięki narzędziom do web scrapingu jesteśmy w stanie analizować kod źródłowy strony i wyciągać nieustrukturyzowane dane.

3.1.2 Wykorzystanie API innych firm

3.2 Przechowywanie danych

3.2.1 Bazy relacyjne

3.2.2 Bazy nierelacyjne

3.2.3 Cache

3.3 Zarządzanie przepływem danych

3.3.1 Data processing

Rozdział 4

Zarządzanie projektem

4.1 Systemy kontroli wersji

W systemach informatycznych występuje bardzo szybkie tempo zmian. Aby wspomóc programistów w zarządzaniu wersjami oprogramowania, wprowadzono systemu kontroli wersji. Najpopularniejszy obecnie jest GIT. Istnieją również platformy internetowe bazujące na tym systemie, które dostarczają interfejs graficzny i umożliwiają łatwiejszą komunikację programistom pracującym nad projektem. Najpopularniejsze platformy to Github, Gitlab czy BitBucket. Są to rozproszone systemy, dzięki temu każdy z programistów ma możliwość pracy w wydzielonym środowisku, nie wchodząc w kolizję innym programistom z zespołu.

4.2 Współpraca na Github

4.3 Zarządzanie zadaniami

4.4 Automatyzacja procesów

Rozdział 5

Projekt aplikacji śledzącej ceny telefonów

5.1 Opis problemu

5.2 Zbieranie wymagań

5.3 Projektowanie interfejsu graficznego

5.4 Programowanie aplikacji z wykorzystaniem frameworka Django

5.5 Architektura aplikacji

5.6 Zbieranie danych do aplikacji

5.7 Uruchomienie aplikacji w chmurze

5.8 Monitorowanie błędów

5.9 Rozwój aplikacji

Rozdział 6

Podsumowanie

Bibliografia

[1] Youtube.

[2] digitallyher.pl. Podstawowe typy aplikacji.

Spis rysunków