Obraz zawierający tekst

Opis wygenerowany automatycznie

**Wydział Informatyki**

**Katedra Metod Programowania**

Programowanie aplikacji biznesowych

**Arkadiusz Dariusz Pańczyk**

Nr albumu s18706

**Aplikacja internetowa wspomagająca monitorowanie i planowanie budżetu domowego**

Praca inżynierska

pod kierunkiem:

dr Krzysztof Barteczko

Warszawa, styczeń, 2022

**Spis treści**

[Wstęp 3](#_Toc87990953)

[1. Zarządzanie budżetem domowym 3](#_Toc87990954)

[1.1. Cel projektu 3](#_Toc87990955)

[2. Koncepcje oraz funkcjonalności systemowe 4](#_Toc87990956)

[3. Rozwiązania technologiczne 5](#_Toc87990957)

[3.1. Backend 5](#_Toc87990958)

[3.1.1. Kotlin 5](#_Toc87990959)

[3.1.2. Spring Boot 7](#_Toc87990960)

[3.1.3. Jpa 7](#_Toc87990961)

[3.2. Frontend 7](#_Toc87990962)

[3.2.1. TypeScript 7](#_Toc87990963)

[3.3. Baza danych 8](#_Toc87990964)

[3.3.1. PostgreSQL 8](#_Toc87990965)

[3.3.2. H2 8](#_Toc87990966)

[3.4. Narzędzia programistyczne 8](#_Toc87990967)

[3.4.1. IntelliJ 8](#_Toc87990968)

[3.4.2. VisualStudio Code 8](#_Toc87990969)

[3.4.3. GitHub 8](#_Toc87990970)

[3.5. Inne rozwiązania technologiczne 8](#_Toc87990971)

[3.5.1. Git 8](#_Toc87990972)

[3.5.2. Docker 8](#_Toc87990973)

[4. Zamysł architektury systemu 9](#_Toc87990974)

[5. Implementacja 10](#_Toc87990975)

[5.1. Moduł rejestracji i logowania 10](#_Toc87990976)

[6. Podsumowanie 11](#_Toc87990977)

Wstęp

Przedmiotem

1. Zarządzanie budżetem domowym
   1. Cel projektu
2. Koncepcje oraz funkcjonalności systemowe
3. Rozwiązania technologiczne

Jako temat aktualnego rozdziału zostały postawione główne technologie użyte podczas procesu tworzenia rozwiązań opisywanej aplikacji internetowej. W poniższych podrozdziałach zaproponowany został podział na mniejsze części, biorąc pod uwagę to w jakim stopniu można je podzielić ze względu na używane w nowoczesnych aplikacjach, warstwy technologiczne.

Wybór języków programowania oraz używanych w procesie budowania aplikacji bibliotek, został wyselekcjonowany ze względu na nowoczesność, ogólnodostępność oraz trendy występujące aktualnie na rynku programistów, oraz polecanych przez nich technik.

* 1. Backend

Warstwa backend, bardzo często nazywana także częścią serwerową aplikacji internetowej, która w praktyce jest całkowicie niezauważalna przez użytkownika końcowego. To właśnie po stronie serwera stoi obsługa całej logiki biznesowej oraz, procesów wymiany informacji wewnętrznych oprogramowania tworzonego dla użytkowników jak i również odpowiednie sprawdzenie czy ze strony klienta aplikacji. Aby nie pojawiały się krytyczne błędy, mogące zagrozić niepoprawnemu funkcjonowaniu aplikacji.

W najbardziej podstawowych aplikacjach, część serwerowa odpowiada również za dostarczenie potrzebnych informacji do bazy danych, która często jest jednocześnie brana jako cząstka całej warstwy backend.

* + 1. Kotlin

Kotlin, czyli powstały w lipcu 2011, statycznie typowany język programowania. Jego pierwsza oficjalnie stabilna wersje wyszła dopiero w lutym 2016 roku. Twórcy języka, czyli firma JetBrains, podczas procesu tworzenia, projektowali go z myślą o pełnym współdziałaniu z innym, bardzo popularnym językiem - Java oraz JVM - jego wirtualną maszyną.

Od późniejszych wersji dostępny jest również Kotlin/JS, czyli łatwe tłumaczenie kodu napisanego, w języku Kotlin na JavaScript, pozwalające na pisanie również frontendu aplikacji bezpośrednio w języku Kotlin. Głównym zamysłem powstania Kotlin/JS było wytworzenie własnej logiki pisania części frontendu aplikacji, jak również części backendowej w przypadku używania Node.js jako głównej technologii użytej do pisania logiki serwerowej[[1]](#footnote-1).

Projektanci języka, trzymali się tego, że Kotlin ma być obiektowym językiem, ukierunkowanym na działanie w środowisku produkcyjnym oraz poprawiającym znane błędy z Javy. Współdziałanie z kodem języka Javy, zostało zachowane, aby w prosty sposób umożliwić firmom, operującym od wielu lat na Javie na stopniową migrację na produkt firmy JetBrains.

Niektóre charakterystyczne cechy Kotlina[[2]](#footnote-2):

Bezpieczeństwo względem błędów z null’ami – w przeciwieństwie do często spotykanej wady języka Javy, bazowo, wszystkie zmienne ustawiane są przez kompilator jako *non-null,* co już częściowo pozwala wyeliminować niechciane wyjątki. Przy pomocy operatora „?.”, Kotlin pozwala na bezpieczne wywołanie metody lub zmiennej. W przypadku, kiedy któraś z wartości okaże się null’em, właściwość nie zostanie wywołana. Pomocny okazuję się równie operator Elvis „?:”, który pozwala zastąpić wartością domyślną, jeżeli przypisywana przez kompilator wartość okaże się null’em.

* Brak możliwości wywołania sprawdzonych wyjątków (Checked Exception) – taka funkcjonalność została całkowicie pominięta, przez twórców Kotlina.   
  W języku Java, używanie podstawowych metod, często wiąże się z przymusowym obudowaniem wywołania instrukcją *try-catch*. Omawiane rozwiązania mogą prowadzić do obniżenia jakości kodu, jak i zaciemnienia rzeczywistego obrazu działania metody.
* Korutyny w programowaniu współbieżnym – Początkowo w języku Kotlin dostępne były wyłącznie, dobrze znane z innych języków wątki. Korutyny, podobnie do wątków pozwalają na obsłużenie, większej ilości procesów jednocześnie. Na korzyść, w przeciwieństwie do wątków, korutyny pozwalają na dynamiczne zarządzanie procesami. Po zawieszeniu działania, uprzednio używany wątek procesora, jest zwalniany i może zostać użyty przez inny proces, co pozwala na bardziej przemyślane zarządzanie wykorzystywanymi wątkami procesora.

Prostota oraz mądrze przemyślane decyzje rozwojowe języka, pozwalają Kotlinowi od wielu lat znajdować się w listach najbardziej dynamicznie rozwijających się języków na świecie. W rozwoju swoją cegiełkę dołożył także Google, który na swojej konferencji I/O w 2017 roku, ogłosił Kotlina oficjalnym językiem programowania Androida – platformy posiadającej 73%, udziału rynku mobilnego na całym świecie[[3]](#footnote-3).

Widząc dotychczasowy rozwój, języka firmy JetBrains, można oczekiwać od Kotlina, że z pewnością będzie dalej piąć się w górę w rankingach popularności wszystkich języków programowania. Jednocześnie wprowadzając dodatkowe możliwości dla deweloperów pracujących na platformie wirtualnej maszyny Javy.

* + 1. Spring Boot

Początkowo całkowity produkt jako Spring Framework, rozwijany był już od października 2002 roku przez Rod Johnson. Stworzony na platformie Java Enterprise Edition jako bardziej przyjazne programiście rozwiązanie, w pisaniu backendu w zaawansowanych aplikacjach internetowych. Głównymi powodami, dla których deweloperzy oprogramowania stawiali na szkielet Spring Framework, to ukierunkowanie pracy programisty na funkcjonalność aplikacji i zmniejszenie nakładów pracy na pisaniu powtarzającego się kodu programu.

Rewolucyjny sposób na pisanie servletów, czyli mniejszych części aplikacji wystawionych na działanie serwera oraz następnie zwracanych zawartość opakowaną w protokół http. W bardzo krótkim czasie pozwolił Spring Framework’owi objąć szczyty popularności i przedstawić go jako niezbędne narzędzie, codziennie używane podczas pisania biznesowych aplikacji działających w obszarze internetowym.

Kolejnym wielkim krokiem w rozwoju szkieletu aplikacji Spring’a był moduł Spring Boot. Powodem, dla którego powstał Spring Boot, był lepszy, ale nadal nie do końca zrozumiały sposób zarządzania projektem. W odświeżonej wersji twórcy próbowali pozbyć się skomplikowanego sposobu zarządzania ziarnami w aplikacji przy pomocy plików XML. W tworzeniu aplikacji webowych, Spring Boot wniósł także funkcjonalności automatycznej konfiguracji zależności oraz mechanicznego wystawiania projektu do web serwera Jetty lub Tomcat, co pozwoliło na możliwość szybszego stawiania aplikacji, a wcześniej wymagało skomplikowanego procesu wdrażania projektu do zewnętrznych plików War.

Diagram

Description automatically generated

Rysunek 1 Pivotal Software; Schemat modułów Spring Runtime; źródło: https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch01s02.html

Moduły Spring Framework:

* Core – Najbardziej kojarzone ze Spring’iem funkcjonalności. Wprowadza do aplikacji popularne właściwości takie jak wstrzykiwanie zależności do ziaren aplikacji jak i języka wyrażeń Spel, pozwalające rozwijać dynamiczne odwołania do klas za pomocą ciągów znaków.
* Data – Zawierający potrzebne, do łączenia się z bazą danych funkcjonalności przy pomocy JDBC lub ORM.
* Web – Moduł wyciągający użyteczność łączenia się użytkownika zewnętrznego przy użyciu protokołu http.
* AOP i Instrumentation – Moduł wprowadzający programowanie oparte na aspektach. Dodatkowo pozwala na wprowadzenie przydatnych serwisów, poprzez instrumentowanie kodu bajtowego do metod, ukierunkowane na gromadzenie danych.
* Test – Moduł zawierający klasy potrzebne, do przetestowania projektu przy użyciu funkcjonalności takich jak testy jednostkowe i integracyjne.

Podstawowym cyklem rozwoju aplikacji, jest używanie ziaren dostępnych w tworzonym programie przy użyciu różnego rodzaju adnotacji, dostępnych bezpośrednio ze Spring Framework lub wyciągania ich prosto z kontekstu aplikacji. Podstawowa funkcjonalność wstrzykiwania zależności pomiędzy klasami i zmiennymi, realizowana jest właśnie przez ziarna Java.

Dobrodziejstwo udostępnione przez tworzenie nowych obiektów, może być realizowane poprzez wzorzec projektowy, wstrzyknięcia zależności, udostępniana właściwość pozwala na oczyszczenie kodu, z powtarzających się wywołań nowych obiektów o tej samej budowie. Zakres tworzenia nowych bean’ów można podzielić na pięć typów:

* *Prototype* – debiutująca instancja, przy następnym wstrzyknięciu
* *Singleton* – taka sama na wszystkie aplikacje
* *Session* – przy łączeniu kolejnej sesji
* *Request* – przy nowym żądaniu http
* *Websocket* – taka sama dla nowego łączenia z WebSocket
* *Application* – taka sama na jedną aplikacje

W Spring’u dostępne są adnotację, oznaczane poprzez „@” oraz słowo opisujące zależność, które pozwalają na dodanie przydatnych funkcjonalności dla klas, jak i zmiennych. Niektóre z przykładowych zależności (każdy poprzedzony „@”):

* Configuration – zależność bazowa, definiująca konfigurację ustawioną poprzez kod.
* Bean – zależność ustawiająca przedstawianą metodę jako ziarno.
* Component – zależność stereotypu, ustawia klasę jako ziarno w kontekście.
* Controller – opisuje klasę jako kontroler potrzebny do WebApi.
* Service – identyfikująca klasę jako implementacja dla logiki biznesowej.
  + 1. Jpa
  1. Frontend

Warstwa frontend to część aplikacji webowej, która pozwala użytkownikowi na swobodną interakcję z oprogramowaniem, najczęściej przy pomocy przeglądarki internetowej. Jest to z reguły, pierwszy możliwy kontakt odbiorcy treści z zaimplementowanym systemem.

Część graficzna reprezentowana jest przez uprzednio utworzony wygląd oraz uzupełniona dostępem do najważniejszych funkcjonalności obsługiwanych przez stronę serwerową, oprogramowania wydaną do obsługi dla użytkowników końcowych aplikacji internetowej.

* + 1. TypeScript

TypeScript pojawił się w roku 2012 jako znacząco rozwinięty JavaScript. Jako główne rozszerzenie względem języka JavaScript można uznać wzbogacenie o dodatkową możliwość dopełnienia informacji o typie zmiennej. Pierwsza, stabilna wersja języka została wydana dopiero po dwóch latach, 12 kwietnia 2014 roku, przez twórców – firmę Microsoft. Swoją rosnącą popularność zawdzięcza w dużej mierze produktowi Angular, przy produkcji którego, jego twórcy, czyli firma Google jako główny język w procesie tworzenia używali właśnie TypeScript. Użyteczność języka wykorzystywana jest również w konkurencyjnym produkcie dla Angulara – ReactJS, gdzie społeczność co raz częściej korzysta z TypeScript jako głównego języka podczas pisania aplikacji internetowych.

Język dobrze został przyjęty, przez społeczność, między innymi ze względu na rewolucyjne zmiany rozwiązujące problemy występujące w JavaScript. W odróżnieniu od innych, stworzonych w przeszłości języków transkompilowanych do JavaScript ’u, nie był wyłącznie wizualną nakładką na składnie kodu, ale także dodawał wysoce znaczące zmiany przydatne w codziennej pracy dla deweloperów oprogramowania.

Główne zmiany w stosunku do języka JavaScript[[4]](#footnote-4):

* Typowanie zmiennych, klas, interfejsów
* Interfejsy
* Typy wyliczeniowe
* Typy generyczne
* Kompilowane wywoływanie

Różnice na korzyść, którą przemawia TypeScript zostały szczególnie dostrzeżone w środowisku korporacyjnym, gdzie Angular wraz z językiem Microsoftu jest wykorzystywany w większości aplikacji, z zamiarem tworzenia części frontendu oprogramowania. Dzięki swojej wewnętrznej właściwości kompilowania się do języka JavaScript, możliwe jest pisanie oprogramowania zarówno po stronie frontendu – Angular czy React, jak również z perspektywy backendu w technologiach takich jak NodeJS.

* + 1. React

Biblioteka frontendowa zaprojektowana i napisana w języku JavaScript, 29 maja 2013 roku przez Jordana Walke, w Facebook’u (aktualnie Meta), wraz z pracującymi tam deweloperami. Miała ułatwiać oraz rozszerzać możliwości jakie dotychczasowo były osiągalne przy tworzeniu interfejsu użytkownika.

Z założenia pozwala na tworzenie aplikacji typu SPA – *Single Page Application*, czyli realizującej założenia aplikacji, ładując dynamicznie zawartość na stronie, jednocześnie bez potrzeby przeładowywania całego serwisu.

Na swoją popularność i renomę React mocno pracuję już od paru lat, początkowo goniąc framework Angular w uwadze użytkowników, aż po wieloletnie królowanie w większości cenionych przez ogólnoświatowych deweloperów oprogramowania rankingów. Różnice w cechach w odróżnieniu do innych frameworków dla frontendu, które zostały dostrzeżone przez użytkowników pozwalają, jeszcze na wiele lat bycia numerem jeden dla React.

Graphical user interface, application

Description automatically generated

Rysunek 2 Porównanie popularności największych technologii framework; źródło: https://trends.google.com

Na powyższej ilustracji łatwo zauważyć dominację React’a w ostatnich latach. Po samej liczbie wyszukań w najpopularniejszej wyszukiwarce – Google, łatwo zauważyć, praktycznie 50% większą popularność względem drugiej najpopularniejszej technologii – Angular.

Proces pisania interfejsu użytkownika w React opiera się na dzieleniu całości aplikacji na mniejsze, niezależne od siebie komponenty wielokrotnego użytku. Celem komponentów jest dostarczenie do kodu w aplikacji dodatkowej funkcjonalności w sposób izolowany, przez swoją logikę, bardzo często zwracający opakowany fragment kodu w HTML.

Komponenty można podzielić na dwa rodzaje:

* Komponenty funkcyjne – Reprezentowane przez prostą funkcję, w teorii rozwijające większą wydajność. Dzięki swojej prostszej budowie i posiadaniu mniejszej ilości zapisanego kodu, również bardziej czytelny, jak i zapewniający łatwiejszy proces debugowania i testowania. Pierwotnie wyróżniał go brak stanu, jednak po wprowadzeniu React Hooks, również ta funkcjonalność została wprowadzona. Do utworzenia prostego komponentu wystarczy podanie mu zmiennej z właściwościami oraz zwrócenie poprawnego fragmentu JSX.
* Komponenty klasowe – Rozwijające funkcjonalności komponentów poprzez dostarczenie zaimplementowanej klasy. Bardziej wymagająca do zaimplementowania niż komponent funkcyjny, React wymaga, aby klasa zawierała wyspecjalizowany konstruktor, dziedziczenie z klasy *Component* jak izwrócenie metody render z ciałem HTML. Główną korzyścią jaką można czerpać z komponentów klasowych są cykle życia aplikacji, takie jak componentDidMount, których implementacja w ciele funkcji, byłaby zdecydowanie dużo cięższa oraz wymagałaby większych starań.

Rozwiązania technologiczne na podstawie których budowy był React oraz język TypeScript, całkowicie pozwalają na wspólne działanie ze sobą. Powody, na korzyść których warto wybrać TypeScript ponad JavaScript, zaczynają się już od opisu właściwości obu języków. Głównym benefitem, TypeScript jest jego pełne statyczne typowanie zależności.

Najszybszą drogą na wdrożenie nowej aplikacji React/TypeScript jest utworzenie jej przy pomocy menadżera pakietów npm oraz odpowiedniego doklejeniu zależności w komendzie utworzenia nowego projektu React „--template typescript”. Podstawową różnicą jaką będzie można zaobserwować przy użyciu zależności o języku TypeScript będą, zupełnie inaczej wygenerowane pliku projektowe, między innymi[[5]](#footnote-5):

* .tsx – wszystkie pliki z rozszerzeniem .js i .jsx należące do JavaScript, zostaną odpowiednio zastąpione plikami TypeScript i TypeScript JSX.
* tsconfig.json – w miejscu pliku jsconfig.json, wyspecjalizowany plik konfiguracyjny zawierający szablon, z domyślnie dostarczonymi ustawieniami projektu.
* react-app-env.d.ts – pozwala w osadzaniu w aplikacji podstawowych zależności z szablonu aplikacji React oraz wprowadzaniu z niego plików.
  + 1. Material-UI
  1. Baza danych
     1. PostgreSQL
     2. H2
  2. Narzędzia programistyczne
     1. IntelliJ
     2. VisualStudio Code
     3. GitHub
  3. Inne rozwiązania technologiczne
     1. Git
     2. Docker

1. Zamysł architektury systemu
2. Implementacja
   1. Moduł rejestracji i logowania
3. Podsumowanie

**Bibliografia**

1. JetBrains; Przyczyna powstania Kotlin/JS https://kotlinlang.org/docs/js-overview.html#use-cases-for-kotlin-js; dostęp 15.11.2021 [↑](#footnote-ref-1)
2. JetBrains; Charakterystyka Kotlina; www.kotlinlang.org/; dostęp 15.11.2021 [↑](#footnote-ref-2)
3. S. O'Dea, 29.06.2021; Statistica.com; Udział różnych platform w rynku mobilnym 2012-2021 www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/; dostęp 15.11.2021 [↑](#footnote-ref-3)
4. Microsoft; Różnice w TypeScript https://www.typescriptlang.org/; dostęp 21.11.2021 [↑](#footnote-ref-4)
5. React; Dodanie TypeScript do aplikacji; https://create-react-app.dev/docs/adding-typescript/; dostęp 21.11.2021 [↑](#footnote-ref-5)