**AKADEMIA MARYNARKI WOJENNEJ**

**im. BOHATERÓW WESTERPLATTE**

**Wydział Mechaniczno-Elektryczny**

Instytut Elektroniki i Automatyki Okrętowej

Katedra Informatyki

**PRACA DYPLOMOWA**

**INŻYNIERSKA**

Temat:

Wykonawca: Adam SZREIBER

Kierownik pracy

dr inż. Jan Masiejczyk

Kierownik katedry

kmdr dr hab. inż. Andrzej Żak

prof. nadzw. AMW

|  |  |  |
| --- | --- | --- |
| Ocena pracy dyplomowej | ………………..…………  słownie | ……………………...……….  data i podpis  Przewodniczącego Komisji Egzaminacyjnej |

Gdynia, dnia …………….…….. r.

Adam Szreiber

21662

**O Ś W I A D C Z E N I E**

Oświadczam, że przedłożoną do egzaminu pracę dyplomową pt. ………………………………………………………………………………………………  
……………………………………………….………………………………………………

kończącą studia I stopnia napisałem samodzielnie. Przy wykonywaniu pracy nie zlecałem jej opracowania ani żadnej jej części innym osobom, jak też nie skopiowałem cudzych opracowań   
i przestrzegałem postanowień Ustawy z dnia 4 lutego 1994 r. o prawie autorskim   
i prawach pokrewnych (Dz. U. z 2018 r. poz. 1191 z późn. zmianami).

Ponadto oświadczam, iż treści zaczerpnięte z literatury przedmiotu są oznaczone w tekście oraz w przypisach, w sposób ogólnie przyjęty dla prac naukowych.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, jestem świadomy(a) zasadności cofnięcia decyzji o wydaniu mi dyplomu.

Wyrażam zgodę na udostępnienie mojej pracy dyplomowej czytelnikom.

Przekazuję moją pracę dyplomową do Ogólnego Repozytorium Prac Dyplomowych.

…………………………………

podpis osoby składającej oświadczenie

Spis treści

[1. Wykaz skrótów i oznaczeń 5](#_Toc97063030)

[2. Wprowadzenie 6](#_Toc97063031)

[3. Dokumentacja projektu 7](#_Toc97063032)

[3.1 dqwedLorem ipsum 7](#_Toc97063033)

[3.2 Lorem ipsum dolor 7](#_Toc97063034)

[4. Implementacja aplikacji 8](#_Toc97063035)

[5. Podsumowanie 9](#_Toc97063036)

[6. Literatura / Załączniki 10](#_Toc97063037)

[7. Streszczenie 11](#_Toc97063038)

# Wykaz skrótów i oznaczeń

|  |  |
| --- | --- |
| JavaScript |  |
| TypeScript |  |
|  |  |

# Wprowadzenie

## Komercyjne aplikacje

Firmy informatyczne wytwarzające oprogramowanie pracują nad swoimi aplikacjami przez kilkadziesiąt lat. Ich code-base może liczyć wiele miloionów linii kodu, rozmieszczone w tysiącach plików. Korporacje starają się zapewnić funkcjonalności jakie oczekują od nich użytkownicy, usprawniają to co kiedyś zostało zrobione, wprowadzają nowe funkcje, tylko po to aby być konkurencyjnym na rynku, Pozyskać nowych użytkownik, utrzymać dotychczasowych dużych, stałych klientów. Na przestrzeni wielu lat struktura, organizacja i pracownicy firmy mogą się zmieniać, a jakość wytwarzanego kodu degraduje się. Aby temu zapobiec każda firma posiada swój styleguild – którego muszą przestrzegać pracownicy firmy. Pomaga to w zachowaniu ogólnego porządku w kodzie programu, ułatwia nowym pracownikom wdrożenie/zapoznanie się z nim, ułatwia proces code-review, ułatwia wyszukiwanie i naprawę błędów. Stosuje się również narzędzia typu SOLARLINT – który kontroluje jakość kodu, a w razie niedostosowania się do zadanych restrykcji informuje o tym np. poprzez podkreślenie zle napisanego kodu i wyświetlenie dokładnego błędu. Firmy stosują również pryncypia które narzucone przez sam język programowania w którym pracujemy. Dlatego tak ważne jest aby styl kodu danego programu był spójny, czytelny i otwarty na rozbudowę.

Napisać trochę o kontrolowaniu wersji… GIT

Wprowadzanie zmian i rozwój produktu jest kluczowym elementem aby zapewnić firmie byt na dynamicznie zmieniającym się rynku oprogramowania. Lecz nie należy to do najłatwiejszych rzeczy gdy pracujemy kodzie napisanym przez innego pracownika lub kodzie który jest bardzo przestarzały – napisany przed wprowadzeniem narzędzi kontrolujących jakość kodu / napisanym przed wprowadzeniem wewnętrznych firmowaych kryteriów kodowania. Aby wprowadzić jakiekolwiek zmiany, najpierw musimy poświęcić wiele godzin aby znaleźć odpowiednie miejsce w kodzie, następnie dokładnie przeanalizować dany fragment kodu, opracować plan wdroż enia nowej funkcjonalności, i ostatecznie przejść do implementacji. Tak złożony proces może przysporzyć wiele trudności, szczególnie mniej doświadczonym programistom.

## Cel pracy

*„Czasu nie cofniesz, ale błędy możesz naprawić.”*

Bezpieczeństow programu to sprawa najwyższje wagi. Firmy budują swoje zaufanie w śród klientów przez dziesiątki lat, jeden krytyczny błąd może spowodować katastrofalne skutki .

LEKKI OPIS TYPÓW BŁĘDÓW APLIKACJI. – BŁASZCZYK.

Serwis Beheer (nid. „zarządzanie”) to projekt który ma wspomóc programisów bezpiecznie wprowadzać zmiany w kodzie progogramu, poprzez feature-flagi, automatyczne wyłączanie flag gdy wystąpi błąd wykonywania kodu, wykresy i statystyki. Wspołczesne programy są niezwykle rozbudowane i nie sposób prześledzić wszystkich ścieżek wykonywanego kodu, co za tym idzie, zmiany w jednym obszarze aplikacji mogą spowodować wystąpienie w innym module aplikacji. Szybkie reagowanie na takie zachowania mogą uratować firmę. Skótki wystąpnienia błędów krytycznych niosa za sobą poważne konsekwencje - w najgorszym przypadku spowodują zamknięcie firmy, sprawy karne, czy zadość uczynienie.. Z historii znamy już kilka takich przypadków:

PODAĆ TUTAJ KILKA KATASTROFALNYCH W SKUTKACH BŁĘDÓW PROGRAMÓW.

Czy wprowadznie narzędzi typu feature-toogle pozwoliłoby tego uniknąć? – Nie jestem w stanie na to odpwiedzeć, ale z pewnością pomogłoby szybciej i precyzyjniej zareagować na te błędy.

## Konkurencyjne rozwiązania na rynku

Feature toggle – launchdarkly

[Feature Flags: Faster software deployment and safer code releases | Split](https://www.split.io/product/feature-flags/)

[The Top 6 Feature Flag Management Tools | Harness](https://harness.io/blog/feature-flag-management-tools/)

[Do czego przydaje się technika Feature Flag (bulldogjob.pl)](https://bulldogjob.pl/news/871-do-czego-przydaje-sie-technika-feature-flag)

Zgłębiając temat Feature-flag można znaleźć wiele komercyjnych produktów na rynku. Niektóre z nich bardzo zaawansowane, zawierające […] złożone wykresy, możliwość serwowania różnych wartości flag na podstawie geolokalizacji, <<<<ITD>>>>

## Wady i zalety feature-flag

ZALETY:

Łatwość przywrocenia poprzedniej ścieżki kodu bez ingerencji w kod

Możliwość automatycznego wyłączania flagi gdy wystąpi błąd

Możliwość serwowania różnych typów flag – wartość string, boolean, intiger

Możliwość wprowadzania większej grupy zmian, kontrolowanej za pomocą 1 flagi

WADY:

Brak dostępu do internetu spowoduje wykonywanie starej ścieżki programu.

Spowolnienie wykonywania programu – komunikacja przez Ethernet jest stosunkowo wolna

Testowanie aplikacji staje się trudniejsze

Brak możliwości wykorzystania flag w high-performance apps

# Budowa Aplikacji

Proces wytwarzania aplikacji jest złożony, na początku ery komputerów, gdy stacje robocze były słabe/wolne i ograniczone przez zasoby, programy pisało się i konserwowało bez większych trudności. Na przełomnie lat 50/60 XIX wieku rozwój technologiczny przyspieszył do takiego stopnia, że zaczęto tworzyć coraz bardziej skomplikowane oprogramowanie, którego implementacja i utrzymanie zaczęło sprawiać trudności.

W latach 1968 i 1969 miały miejsce dwie konferencje w Rzymie i Garmich na których po raz pierwszy wprowadzono termin „INZYNIERIA OPROGRAMOWANIA” (ang. Software engineering, SE) dyscyplina informatyczna stosująca podejście inżynierskie do tworzenia oprogramowania: od analizy i określenia wymagań, przez projektowanie i wdrożenie, aż do ewolucji gotowego oprogramowania.”

Inzynieria oprogramowania zajmuje się metodami wytwarzania, oceniania i utrzymywania oprogramowania systemów komputerowych oraz metodami zarządzania realizacją projektów informatycznych. Celem stosowania tych metod jest zapewnienie wysokiej jakości oprogramowania oraz doprowadzenie do terminowej i zgodnej z budżetem realizacji projektu. Znaczenie metod inżynierii oprogramowania rośnie wraz z wielkością projektu

Dziedzina ta wprowadza wiele procesów (Proce s jest zbiorem aktywności wykonywanych z myślą o osiągnięciu pewnego celu) które należy dobrać indywidualnie do tworzonego projektu. Do realizacji niniejszego projektu użyto modelu kaskadowego (eng. Waterfall).



Jest to szeroko stosowany model charakteryzującym się jasno uporządkowanym przebiegiem pracy, brakiem powtarzania aktywności oraz posiada wbudowany mechanizm weryfikacji wyników każdej z faz. Każda faza następuje po sobie a błąd w jakiejkolwiek faz jest bardzo kosztowne w usunięciu, ponieważ skutkuje powrotem do pierwszej fazy czyli określenia wymagań.

## Określenie wymagań

<<<<<CO TO JEST>>>>>

### Wymagania funkcjonalne

|  |  |
| --- | --- |
| NAZWA FUNCKCJI |  |
| OPIS |  |
| DANE WEJSCIOWE |  |
| POWÓD |  |

|  |  |
| --- | --- |
| NAZWA FUNCKCJI | Rejestracja |
| OPIS | Funkcja pozwala stworzyć konto nowe użytkownika |
| DANE WEJSCIOWE | Adres email, hasło |
| POWÓD | Aby korzystać z serwisu Beheer użytkownik musi posiadać konto |

|  |  |
| --- | --- |
| NAZWA FUNCKCJI | Logowanie |
| OPIS | Funkcja pozwala określić tożsamość użytkownika i przydzielić mu odpowiednie zasoby |
| DANE WEJSCIOWE | Adres email, hasło |
| POWÓD | Aby moc korzystać z serwisu należy być zalogowanym |

### Wymagania niefunkcjonalne

- Aplikacja ma być dostępna 24/7

- Wygląd aplikacji powinien być przyjazny użytkownikowi

- Wdrożenie w system nie powinien trwać więcej niż 15min

- Interfejs graficzny powinien być wyraźny i czytelny

- Interfejs graficzny powinien być w stonowanych kolorach

- Interfejs graficzny pewien mieć ciemną szatę graficzną

- Informacje o błędach powinny być umieszczone w widocznym dla użytkownika miejscu

- Stworzenie nowej flagi nie powinno zajmować więcej niż 1min

- Hasła w bazie danych powinny być przechowywane w bezpieczny sposób

Cos tu będzie ………………………………………………………………………..

## Analiza i Projektowanie

Trochę o UML

Cos tu będzie ………………………………………………………………………..

### Wstęp

*Program komputerowy* to sekwencja instrukcji składająca się z deklaracji i instrukcji, która jest zgodna z zasadami określonego języka programowania w celu przetworzenia i rozwiązania pewnych funkcji, zadań lub problemów za pomocą komputera.

Program realizowany jest jako sekwencja maszynowa, tj. polecenia procesora przetwarzane są przez procesor lub procesory komputera, które są następnie wykonywane. Przez program komputerowy rozumie się również kod źródłowy programu, z którego w trakcie tworzenia oprogramowania tworzony jest plik wykonywalny.

Proces tworzenia programu komputerowego nazywamy programowaniem. Programista do procesu wytwarzania programu komputerowego używa specjalistycznych narzędzi jakimi są języki programowania.

Na świecie istnieją tysiące języków programowania i każdego roku powstają nowe. Od języków naturalnych odróżniają się wysoką precyzją oraz jednoznacznością. Człowiek podczas komunikacji między sobą stale popełnia niewielkie błędy lub pozostawia niedomówienia wiedząc, że drugi rozmówca najczęściej go zrozumie. Maszyny wykonują zadania dokładnie, dlatego każdą czynność trzeba opisać ściśle krok po kroku, ponieważ komputer nie potrafi domyślić się, co programista miał na myśli.

Dla programisty język programowania to swoiste narzędzie do komunikacji z komputerem. Dobierając język programowania powinniśmy zwrócić uwagę na takie aspekty jak: wielkość i złożoność projektu, szybkość działania, live-time projektu, niezawodność, doświadczenie programisty, system operacyjny. Tak więc nie ma języka idealnego – warto, by programista znał ich kilka i zgodnie z zasadą „use the right tool for the right job” oraz własnymi preferencjami wybierał język adekwatny do danego zadania. Zazwyczaj doświadczeni programiści to poligloci, bardzo dobrze znają i posługują się wieloma językami programowania. W większości języków programowania występują te same podstawowe mechanizmy, których używa się w bardzo podobny sposób niezależnie od wybranego języka programowania, a różnice często sprowadzają się wyłącznie do nazw bibliotek, funkcji, klas itp.

Doskonałym tego przykładem jest niniejsza praca w której wykorzystano dwa języki programowania, C# w którym wykonano bibliotekę do wprowadzania nowych funkcjonalności, oraz JavaScript i TypeScript w którym wykonano serwis internetowy Beheer.

### Technologie i biblioteki użyte do realizacji projektu

Technologie:

* PostgreSQL

Darmowa, open-source relacyjna baza danych znajdująca się w top 4 najczęściej wyberanych baz danych. Do głównych jego zalet należą

**Jak oceniano systemy bazodanowe w tym rankingu?**

Podstawą była aktualna popularność mierzona:

* ilością rezultatów w wyszukiwarkach internetowych (Google i Bing),
* ogólnym zainteresowaniem systemem mierzone w Google Trends,
* częstością występowania w dyskusjach technicznych (Stack Overflow i DBA Stack Exchange),
* liczbą ofert pracy, w których dany system był wymieniany (Indeed i Simply Hired),
* liczbą profili w sieciach profesjonalnych, gdzie system był wzmiankowany (LinkedIn),
* liczbą wzmianek w sieciach społecznościowych (Twitter).

1. PostgreSQL jest znany i bardzo wysoko ceniony za stabilność i szeroki zestaw funkcji.
2. PostgreSQL to baza nie tylko relacyjna, uwzględnia wiele rozwiązań post-relacyjnych np. typy danych: JSON, XML, wieloelementowe tablice (arrays), typy wierszy (row types), pełnotekstowe, szeregi czasowe, rekurencyjne zapytania grafowe, dane przestrzenne (GIS); wraz z funkcjami operującymi na tych nowych typach danych i – co więcej – z możliwością tworzenia zapytań mieszanych tj. obejmujących dane relacyjne i post-relacyjne.
3. PostgreSQL używa standardowego SQL bez specyficznych dodatków utrudniających przeniesienie kodu lub przestawienie się programisty z innej bazy danych (ograniczenie vendor lock-in).
4. W ciągu ostatnich lat kolejne wydania PostgreSQL skupiały się na wzmocnieniu back-endu, silnika bazy danych, dzięki czemu uzyskano istotny wzrost wydajności i skuteczności.
5. Nowe cechy funkcjonalne i użytkowe są idealne dla wdrożenia środowiska DevOps.
6. Migracja do PostgreSQL jest łatwa, a dzięki temu atrakcyjna dla wielu użytkowników Oracle, zmagających się z polityką cenową i licencyjną tej firmy.
7. PostgreSQL ma [najbardziej wolnościową i o ile można to wiarygodnie ustalić najkrótszą licencję](https://www.postgresql.org/about/licence/) spośród wszelkich licencji open source, co skutkuje także [szeroką popularnością wśród społeczności deweloperów](https://github.com/search?q=postgres&type=repositories) i co za tym idzie także [szeroką gamą sponsorów](https://www.postgresql.org/about/sponsors/).
8. PostgreSQL ma jedną z najszerszych ofert wsparcia komercyjnego wymaganego przez przedsiębiorstwa dla rozwiązań produkcyjnych w tym [najpopularniejsze w Europie EDB i 2ndQuadrant](https://www.postgresql.org/support/professional_support/europe/). Rozwiązania EDB posiadają wszystkie zalety bazy danych PostgreSQL, wzbogacając ją o zaawansowane narzędzia do migracji, integracji i zarządzania, a także funkcjonalności klasy enterprise oraz support 24/7.

* Node.js

Node.js to wieloplatformowe, asynchroniczne, sterowane zdarzeniami srodowisko uruchomieniowe dla języka JavaScript. To oznacza, że za jego pomocą można uruchomić kod JavaScript bezpośrednio na maszynie lokalnej. Domyślny manager pakietów NPM sprawia, że deweloperzy łatwo zarządzają bibliotekami i zależnościami w projektach, a ogromna baza bibliotek ułatwia tworzenie skomplikowanych aplikacji. Język programowania jakim jest JavaScript odznacza się niskim progiem wejścia, przez co w ostatnim czasie platforma zyskała ogromną popularność. Node.js tworzony był z myślą o szybkim budowaniu skalowalnych aplikacji internetowych.

* C# .Net

To bezpłatna platforma deweloperska, którego językiem programowania jest między innymi C#. Pozwala tworzyć oprogramowanie takie jak:

- aplikacje internetowe

- mikrousługi

- aplikacje desktopowe

- aplikacje mobilne

- gry

- aplikacje konsolowe

- internet rzeczy (IoT)

Do głównych zalet .Net należą:

- wieloplatformowość

- może być uruchamiany na wielu architektach procesorów

- open-source

- jeden code-base

- bezpieczeństwo wynikające z typowanie danych

- dedykowany manager pakietów .Nuget

- narzędzie .Net interactive – pozwalający na wykonywanie fragmentów kodu w konsoli

- kompilator JIT (Just in time compiler) – kompilacja kodu odbywa się podczas wykonywania aplikacji, w ten sposób można ograniczyć czas kompilacji ogromnych i złożonych systemów

- Garbage collector - Automatyczne zarządzanie pamięcią

- Linq – technologia pozwalająca posługiwać się składnią przypominającą zapytania SQL odnosząc się do źródła danych jakim jest tablica w kodzie programu.

Biblioteki i frameworki

* WinForms

Windows Forms (często nazywana w skrócie WinForms) jest biblioteką wizualną, umożliwiającą zaprojektowanie graficznego interfejsu użytkownika (GUI, czyli Graphical User Interface). WinForms jest częścią środowiska .NET Framework. Windows Forms jest zbiorem elementów służących do tworzenia aplikacji okienkowych.

Aplikacje budowaną w oparciu o formatki nazywamy sterowaną zdarzeniami. Takowa aplikacja przez większość czasu oczekuje na czynności użytkownika programu np. wciśnięcie przycisku czy wypełnienie pola formularza.

* Blitz js

To framework inspirowany na Ruby on Rails przeznaczony dla programistów full-stack, bazuje na innym JavaScriptowym frameworku next.js. Tworząc nowy projekt w Blitz otrzymyjemy na start wiele domyślnych rozwiązań które mają ułatwić pracę programiście i sprawić, że zajmiemy się pracą nad rozwijaniem projektu a nie budowaniem i przygotowywaniem. Jest to możliwe dzięki wbudowanym rozwiązaniom takim jak:

- zero-api data layer

- autentykacja

- autoryzacja

- domyślna struktura podziału plików

* Faker

Biblioteka do generowania ogromnych ilości losowych, ale wciąż realistycznych danych. Stosowana między innymi w celach wypełnienia baz danych przykładowymi danymi, testowania aplikacji, przygotowywania wersji demo aplikacji. Posiada dobrze udokumentowany i czytelny interfejs API, a generowanie danych następuje po wywołaniu odpowiedniej metody.

* ChakraUi

Jest to prosta, modułowa, udostępniająca gotowe komponenty biblioteka, która pozwala na budowanie wyglądu aplikacji w sposób blokowy. Posiada domyślnie zaimplementowane style dla udostępnianych komponentów, co sprawia, że korzystając podstawowych komponentów aplikacja będzie wyglądać czytelnie i schludnie.

* Nodemaier

To moduł dla aplikacji budowanych w srodowisku Node.js umożliwiający łatwe wysyłanie wiadomości e-mail. Autorzy tej biblioteki stawiali dużą wagę na bezpieczeństwo, co przekłada się ogromną popularność i wszechstronność. Moduł udostępnia szereg funkcjonalności takich jak: wysyłanie emalii jako zwykły tekst lub dokumenty html, dodawanie załączników, autentykacje OAuth2, bezpieczne dostarczanie emaili za pomocą protokołu TSL. Bibioteka nie jest zamknięta, oznacza to ze można tworzyć dodatkowe wtyczki.

* Uuid

Moduł pozwalający tworzyć indentyfikatory UUID (en. Universally Unique Identifier). Jest to ciąg 128 bitów, który gwarantuje unikalność w przestrzeni I czasie. Stosowany często w ścieżkach URL.

* React

React jest deklaratywną, wydajną i elastyczną biblioteką javascriptową do budowania interfejsów użytkownika. Pozwala na tworzenie złożonych UI przy użyciu małych i odizolowanych od siebie kawałków kodu, zwanych “komponentami”.

React niesie ze sobą wiele korzyści, takich jak reużywalność komponentów, witralne drzewo DOM, jednokierunkowy przepływ danych, lekkość i stabilność, niski próg wejścia, łatwość w przerzuceniu się na aplikacje mobilne.

React znacznie ułatwia tworzenie interaktywnych UI. Przezaczony do ptojektowania prostych widoków obsługujących stan aplikacji, React domyślnie zajmie się sprawną aktualizacją i ponownym renderowaniem odpowiednich komponentów.

Deklaratywne widoki sprawiają, że kod staje się bardziej przewidywalny i łatwiejszy do debugowania.

* Prisma

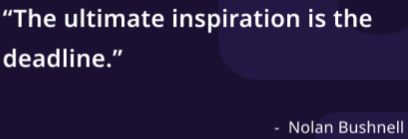
Prisma to biblioteka przeznaczona dla programistów backend ułatwiająca odczytywanie i zapisywanie informacji do bazy danych w intuicyjny i bezpieczny sposób. Wspiera zarówno relacyjne jak i nierelacyjne bazy danych. Najpopularniejsze z nich to, PostgreSQL MySQL SQL Server MongoDB.

Należy wspomnieć, domyślnie otrzymujemy również narzędzie o nazwie „Prisma studio” które umożliwia edytowanie i przeglądanie danych w graficznej formie – dane wyświetlane są w formie tabel.

* Chart js

Darmowa, oparta na elemencie <canvas> html, bliblioteka open-source napisana w języku JavaScript, przeznaczona do wizualizacji danych. Tworzenie wykresów jest łatwe i elastyczne. Polityka opensource sprawia, że biblioteka wciąż dynamicznie się rozwija, zdobywając popularność i rzeszę stałych użytkowników. Natomiast dostępny system wtyczek to najbardziej efektywny sposób aby spersonalizować lub zmienić domyślne zachowanie wykresów. Bibilioteka wspiera takie wykresy jak: wykres warstwowy, liniowy, słupkowy, bąbelkowy, punktowy, kołowy, typu radar.

## Implementacja



Po całym żmudnym I czasochłownnym, ale jakże ważnym procesie planowania systemu przychodzi czas na fazę implementacji, w którym to tworzony jest gotowy produkt. Faza implementacji to jedna z najbardziej kosztowych faz wytwarzania oprogramowania. W firmach zajmujących się tworzeniem oprogrmowania pochłania na ogól dwie trzecie zasobów i połowę czasu trwania projektu. Duża koncentracja zasobów w stosunkowo krótkim czasie wynika z możliwości równoległego prowadzenia prac nad wieloma komponentami przez niezależnych ludzi lub zespoły. Faza konstrukcji jest zwykle okresem największego wysiłku i najwyższego zatrudnienia po stronie wykonawcy.

Autor pisząc pracę osobiście nie zważa na takie czynniki jak koszt, daje to częściową swobodę, natomiast ograniczony czas, przysparza problemów nie tylko autorowi ale i całym zespołom programistycznym. Sytuację w której zbliża się deadline oddania gotowego produktu, a system wciąż wymaga nakładu sił i czasu programistów nazywamy „Crunch mode”. W takiej sytuacji jest kilka wyjść:

- nadgodziny

- skupienie jako pierwsze się na najbardziej znaczących funkcjonalnościach z perspektywy zleceniodawcy

- przydzielenie doświadczonych zespołów do trudniejszych zadań

- stworzenie prototypowego rozwiązania, a następnie stopniowe nadbudowywanie do wersji w pełni funkcjonalnej

- używanie zewnętrznych bibliotek zamiast tworzenia własnych rozwiązań

- zwiększenie liczebności drużyn pracujących nad danym projektem

- tymczasowe przydzielenie doświadczonego programisty do mniej doświadczonej drużyny znacząco zwiększa produktywność i szerzenie wiedzy

- dokładne określenie **„kamieni milowych”** – głównych celów do osiągnięcia

Projekt Beheer zakłada stworzenie aplikacji internetowej oraz udostępnienia biblioteki dla deweloperów umożliwiającej wprowadzanie w bezpieczny sposób funkcjonalności.

### Aplikacja internetowa

Aplikację internetową stworzono w node.js, a głównym środowiskiem programistycznym był Blitz.js.

Podstawowa struktura folderów aplikacji wygląda następująco:



Każdy z folderów jest integralną, zawiera kompleksową część systemu. Podział pomaga w łatwym nawigowaniu, a także wprowadzając jakiekolwiek zmiany, można być pewnym, że reszta modułów nie zmieni zachowania ani sposobu swojego działania.

* Struktura bazy danych

DIAGRAM

OPIS DLACZEGO TAK PODZIELONE

* Katalog „api” zawiera pliki odpowiadające za komunikację pomiędzy biblioteką programisty a systemem. To w nim zachodzi odpytanie serwisu o wartość flagi, odebranie wyniku z uruchomienia kodu, automatyczne przełączenie flagi w stan false czy wysłanie powiadomienie email.

Moduł ten wymienia z dołączoną biblioteką 2 struktury danych:

|  |  |
| --- | --- |
| Struktura pobierana przez bibliotekę | Struktura danych odbierana przez serwis. Zawiera informacje o rezultacie wykonywanego kodu. |

Aby lepiej zobrazować działanie tej części systemu przygotowano poniższy diagram aktywności.



Funkcja automatycznego przełączania flagi w przypadku wystąpienia błędu przy uruchomieniu nowo zaimplementowanego kodu, pozwala na szybkie reagowanie deweloperów. Z punktu widzenia dużych firm bezpieczeństwo i niezawodność jest najważniejsza, co za tym idzie, lepszym rozwiązaniem jest wycofanie wadliwej funkcjonalności, aby nie ucierpiało zaufanie klientów do firmy. Kolejnym powodem dla którego zdecydowano się wprowadzić ten mechanizm jest często spotykana sytuacja w której plikacja się crachuje, lub wyświetla alert zawierający informacje o błędzie. Używając tej biblioteki taka sytuacja będzie maiała miejsce tylko raz – dla pierwszego użytkownika który odkryje dany błąd w systemie.

Z powyższego diagramu wynika, że gdy flaga nosi wartość false, to nie jest sprawdzana zawartość pola errors obiektu ExecutionResult. Wynika to z **natury**, pierwotny kod powinien być niezawodny. Nie miej jednak wciąż gromadzimy dane w bazie danych. Pozwala to na dalsze udoskonalanie kodu np. gdy luka w nowej funkcjonalności wymaga dużego nakładu czasowego programistów, warto wtedy naprawić podstawowy kod.

* Katalog „auth” zawiera pliki odpowiedzialne jest za logowanie i rejestrowanie nowych użytkowników.

Logowanie i rejestracja to skomplikowany proces, od programisty zależy bezpieczeństwo systemu, a tutaj są przekazywane sensitive dane. Wyciek takich danych niesie za sobą szereg nieporząanych sktków, począwszy od spamu na poczcie email, az po włamania na konta urzytkowników na innych platformach w sieci.

Ze strony użytkownia ważne jest aby hasła do każdego konta nie powtarzały się, posiadały co najmniej 8 znaków, zawierały co najmniej 1 znak specjalny i cyfrę. To zapewnia podsawowe bezpieczeństwo.

Ze strony programisty należy wprowadzić wymienione wyżej ograniczenia co do jakości wprowadzanego hasła. Nie należy również zapisywać haseł do bazy danych w postaci zwykłego tekstu. Zamiast tego należy wprowadzić mechanizm haszowana hasła. Zapewnia to bezpieczeństwo na poziomie przesyłania danych przez sieć, a także potencjalnych wycieków z bazy danych. W ten sposób atakujący zamiast typowego hasła będzie widział ciąg znaków (o długości 128 znaków) , a rozszyfrowanie jest niezwykle czasochłonne. Zazwyczaj wystarcza to aby zniechęcić potencjalnego włamywacza do czynienia zła.

Mechanizm haszowania został zaimplementowany przy użyciu biblioteki secure-password. Bibilioteka ta wykorzystuje algorytm kryptograficzny Argon2id, który jest parametryzowany poprzez: tekst, salt, wyznaczone zużycie pamięci, czas wykonywania algorytmu, ilość wątków, wynikowa długość ciągu znaków. Algorytm został zaprezentowany na zawodach Secure Password Hashing w 2019 , co dodatkowo potwierdza jego bezpieczeństwo.



[funkcja realizująca rejestrację - wykorzystanie biblioteki SecurePassword i proces haszowania]

W projekcie zastosowano bibliotekę ZOD, wspomagającą walidowanie danych, zapewnia to, że programista otrzyma dokładnie takie dane jakich się spodziewa. Korzystając z zod tworzymy chain-method, z perspekrywy dewelopera, jest to niezwykle czytelny i wygody rodzaj api. Zastosowanie tejże biblioteki ukazano na rysunku XXX.



Wyswietlanie błędów przedstawiono na rysunku XXX. Funkcjonalność tą zrealizowano za pomocą komponentu Alert udostępnionego przez bibiotekę ChacraUi.



W projekcie zaimplementowano mechanizm automatycznego ukrywania błędu. Domyślnie wygaśnięcie następuje po 10 sekundach.

* Folder „components”

Zawiera komponenty stworzone przez autora, na potrzeby implementacji systemu.

Koncepcyjnie, komponenty są jak javascriptowe funkcje. Przyjmują one arbitralne wartości na wejściu (nazywane “właściwościami” (ang. props)) i zwracają reactowe elementy opisujące, co powinno się pojawić na ekranie.

Komponenty są reużywalne, dlatego przed ich stworzeniem, warto się zastanowić co dokłanie będziemy prezentować i jakie parametry przekazać, aby później móc je zastosować kolejny raz. Przykładem takiego komponentu może być zastosowany do wyświetlania błędów ErrorViewComponent ukazany na rysunku XXX



Dzięki komponentom możemy budować skomplikowane strony internetowe jak z klocków, reużywalność pomaga w zmniejszeniu ilości kodu aplikacji. Dodatkowo gdy chcemy wprowadzić zmianę w jakimś elemencie witryny, zmieniamy w jednym miejscu w kodzie programu, a efekt będzie widoczny w każdym miejscu wystąpienia komponentu. Podział na komponenty sprzyja także szybkiemu poruszaniu się w projekcie i wyszukiwaniu dokładnie tego co nas interesuje.

React domyślnie udostępnia szereg hooków, które umożliwiają dynamiczną zmianę kontentu zawartego na stronie internetowej. Najczęściej używanym jest hook „useState”, który przechowuje zapisaną wartość, aż do usunięcia danego komponentu z drzewa DOM, co może nastąpić np. po opuszczenia witryny czy przejściu do innej podstrony. Dobrym przykładem ukazaującym wykorzystanie hooka useState jest, rysunek XXX, w którym to tworzony jest widok listy funkcjonalności, na podstawie tablicy ‘features’ - za każdym razem gdy opuścimy witrynę chcemy na nowo pobrać tablicę wszystkich funkcjonalności użytkownika.



* Moduł „hooks”

System do budowy wymagał zastosowania następujących hooków

- useState

- useMutation

- useRouter

- useRouterQuery

- useParm

- useCurrentUser

- useQuery

- useEffect

- useClipboard

- useSession

Jakich potrzebowałem w aplikacji

* Folder Layout

Komponent w nim zawarty opakowuje całą stronę, uzupełniając jedynie sekcje head dokumentu html.

* „models”

Autor tworząc ten projekt zdecydował się użyć specyficznej odmiany języka Javascript jakim jest TypeScript. Jest to jezyk obiektowy, opierający się na Javascript, którego główym atutem jest typowanie zmiennych. Tworząc duże projekty należy zwaracać szczególną uwagę na kod który wytwarzamy, mysi być on czytelny i zrozmiały nie tylko dla autora. Wprowadzając typy danych znacznie łatwiej jest się odnaleźć w kodzie, zotaje zachowany pewien ład, zmiejszona zostaje możliwość popełnienia błędu przez programistę. W TypeScript otrzymujemy wszystkie atuty programowania obiektowego, takie jak dziedziczenie, interfejsy, polimorfizm. Wybierając ten język programowania, nie jest równoznaczne z całkowitym porzuceniem Javascript, wręcz przeciwnie, to włanie stosowanie obu tych języków naprzemiennie stanowi prawdziwą potęgę.

Katalog models zawiera modele danych używanych w kodzie żródlowym aplikacji. Programowanie staje się znacznie łatwiejsze gdy wiemy dokładnie jakie pola zawiera obiekt. Na rysunku XXX zamieszczono model obiektu Feature. Tworząc w aplikacji nową flagę tak naprawdę tworzymy obiekt typu Feature, który kolejno zostaje zapisany do bazy danych. Jak później będzie można zauważyć niewiele się on róźni od struktury tabeli w bazie danych.



Dzięki zastosowaniu modeli mamy podstawową walidację danych – nie możemy przypisać wartości „abc” do zmiennej typu „number” – już w procesie kompilacji otrzymamy błąd.

Kolejnym atutem tworzenia modeli obiektów jest możliwość stworzenia statycznej metody wewnątrz obiektu, który zwróci instancję obiektu w raz z wypełnionymi losowymi danymi. Ułatwia to proces tworzenia oprogramowania, gdy potrzebujemy przetestować zachowanie systemu przy zadanych parametrach. Jest to popularna i często stosowana praktyka.

Zastosowano ten mechanizm w obiekcie PostExecutionData, co przedstawiono na rysunku XXX. Autor wykorzystał ten mechanizm do wypełnienia bazy danych.



* Moduł „mutations”

Katalog mutations, przechowuje wszystkie funkcje użyte w systemie do modyfikacji danych przechowywanych w bazie danych.

* Moduł „queries”

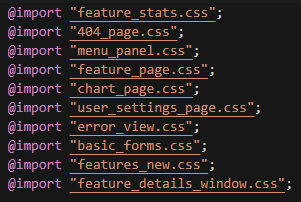
Katalog queries, przechowuje wszystkie funkcje urzyte w systemie do pobierania danych z bazy danych

* Moduł „styles”

Tworząc witreynęinternetową autor zdecydował się użyć dedykowanej bibliteki do tworzenia interfejsów graficznych, nie mniej jednak nie zotał on całkowicie zwolniony z stosowania kodu css. Aby osiągnąć oczekiwany rezultat, należało wyedytować domyślne style komponentów, lub stworzyć własny komponent i nadać mu odpowiednie cechy wyglądu. Do tego służą właśnie kaskadowe arkusz stylów – w skrócie CSS.

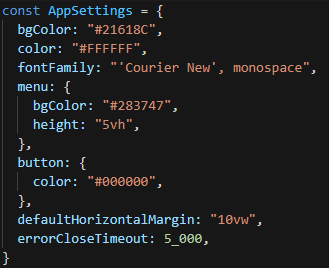
Podział jaki zastosowano spławia że każdy z plików zawiera nie więcej niż 100 lini, przez co odnalezienie się i edycja nie sprawia większego kłopotu. Starano się stworzyć odzienly plik css dla każdego głównego komponentu reactowego. Należy wspomnieć o pliku 404\_page.css którego zawartość pobrano i wdrożono z darmowego zródła.





Z myślą o rozszerzalności autor zastosował jeden główny plik sheet.css w którym imporuje wszystkie pozostałe pliki css. W pliku sheet.css znajduje się także ostylowanie elementów bazowych, które widoczne są na wszystkich widokach witryny.

* Plik Settings.ts



Myśląc o dalszym rozwoju aplikacji sporządzono globalny plik konfiguracyjny całej aplikacji nazwany - settings.ts

* Moduł „lang”

System został zaprojektowany, w taki sposób, aby nie ograniczać się tylko do rynku polskiego. Dlatego tez powstał moduł lang, w którym znajdują się tłumaczenia wszystkich widocznych napisów zamieszczonych w interfejsie użytkownika.

* Pages



Katalog pages, przechowuje pliki stron internetowych. Na rysuknku XXX ukazano strukturę w jakiej są one przechowywane, nie jest to przypadkowe ustawienie, ponieważ specyficzną cechą frameworka Blitz.js jest tak zwane dynamiczne trasowanie (eng. Dynamic Routes). Jest to mechanizm który konwertuje fizyczną ścieżkę do pliku na ścieżkę url. Przykładem może być witryna przedstawiona na rysunku XXX. Ścieza url to /about, która prowadzi do pliku index.tsx zawartego w folderze about.



Główną założeniem projektu Beheer jest tworzenie i zarządzanie funkcjonalnościami. Jako iż użytkownicy będą spędzać w tej części aplikacji, interfejs użytkownika powinien być czytelny i intuicyjny. Autor zdecydował się aby listę funkcjonalności przedstawić w tabeli, jak ukazano na rys XXX. Wybór tego rozwiązania pozwolił na umieszczenie przycisków kopiuj – zielony z ikoną dwóch kartek nakładających na siebie, przycisk typu swich – odpowiadający za przełączanie stanu flagi, oraz przycisk usuń – czerwony opatrzony w ikonę kosza. W odróżnieniu od tekstu stosowanie ikon przykuwa uwagę użytkownika, zajmują mniej miejsca na stronie – szczególnie ważny aspekt tworząc witrynę przeznaczoną na urządzenia mobilne, łatwo je ostylować i wdrożyć, urozmaicają treść – nie zlewają się z resztą strony.

Na górze strony można zauważyć tekst „Jeśli nie ma tutaj Twojej funkcjonalności – stworz ją.”. Jest to odnośnik który przenosi użytkownika do podstrony z formularzem tworzenia funkcjonalności – wymagane jest tylko wprowadzenie nazwy nowej funkcjonalności.

Biorąc pod uwagę obszerne systemy deweloper może chcieć tworzyć dziesiątki, jeżeli nie setki flag. Aby zwiększyć komfort korzystania z serwisu i zaoszczędzić czas użytkowników, autor zdecydował się wprowadzić pole wyszukiwania.



Implementację wyszukiwania funkcjonalności zaprezentowano na rysunku XXX.

Wspomniano wcześniej, że lista obiektów Feature przechowywana jest w stanie komponentu FeatureList, za każdym razem gdy następuje przeładowanie strony lista dostępnych funkcjonalności zostaje zaciagnięta z bazy danych i zapisana do zmiennej features. Podczas wyszukiwania nie możemy stracić pierwotnej listy funkcjonalności, a zaciąganie jej na nowo za każdym razem gdy użytkownik wprowadzi tekst do pola wyszukiwania byłoby zbyt powolne i tworzyło niepotrzebne zapytania do bazy danych. Zdecydowano się utworzyć kopię pełnej listy funkcjonalości – którą nazwano allFeatures. Inicjalizowana jest w momencie wpisywania tekstu w polu wyszukiwania. Kolejno podstawie wartości jaką wpisał użytkownik zapada decyzja, jeśli pole jest puste – należy wyświetlić wszystkie funkcjonalności, w przeciwnym wypadku należy wybrać tylko te funkcjonalności których nazwa rozpoczyna się od wprowadzonego tekstu. Dla wygody użytkownika wyszukiwanie nie uwzględnia wielkości liter (case insensetive).



Klikając na funkcjonalność w tabeli z rysunku XXX, zostaniemy przeniesieni do widoku wykresu, przedstawiono to na rysunku XXX. Graficzne przedstawienie danych znacznie lepiej wpływa na odbiorcę niżeli czyste dane. Wykres domyślnie sporządany jest z ostatnich 30 dni. Autor aby zachować przejrzystość i z myślą o dalszym rozwoju, zbudował obiekt DataAdapter, który przyjmuje tablicę obiektów PostExecutionData – obiekty te otrzymujemy w wyniku uruchomienia kodu.

Wewnątrz obiektu DataAdapter, odbywa się szereg kalkulacji. Korzystając z metod udostępnionych przez interfejs api obiektu, który przedstawiono na rysunku XXX, otrzymujemy dane potrzebne do sporządenia wykresu.

<<<PRZEDSTAWIĆ SCREEEN - INTERFACE DATA\_ADAPTERA>>>



* katalog „utils”

W katalogu utils zamieszczone są głównie funkcje które wspomagały autora w tworzeniu oprogramowania. Przykładem takiej funkcji może być RandomInt(), która zwraca losową liczbę całkowitą z przedziału sprecyzowanego w parametrach funkcji.

Tworzenie takich funkcji bezpośrednio w pliku z komponentem, znacznie zwiększałoby ilość linii kodu, a importowanie do kolejnych komponentów, wpływałoby negatywnie na jakość i czytelność kodu.

* Moduł „db”

Jeżeli chcemy gromadzić i składować dane to nieodłączną częścią systemu jest baza danych. W bazach danych dane przechowywane są w tabelach, dzięki temu są usystematyzowane, co usprawnia ich przetwarzanie i modyfikowanie.

Każda tabela posiada nazwę oraz szereg atrybutów, co odzwierciedla model w programowaniu obiektowym. Na rysunku XXX zaprojektowano bazę danych wykorzystaną w niniejszym projekcie.

<<<DB DIAGRAM>>>

Zazwyczaj do komnikacji z bazą danych używany jest specjalny język SQL, natomiast Blitz.js przychodzi z narzędziem zwanym Prisma. Prisma jest bibilioteką która wspomaga programistę w budowaniu bazy danych poprzez deklarowanie schematów. Ułatwia także wykonywanie zapytań – programista dostarcza pewną strukturę która następnie konwertowana jest na zapytanie SQL. Zmniejsza to ilość błędów popełnianych przez programistę – zaawansowane zapytania SQL potrafią być bardzo obszerne, nie sposób stworzyć zapytanie pobierające z więcej niż jednej tabeli bez pomyłki, a także umożliwia łatwą zamianę na inną relacyjną bądź nierelacyjną bazę danych nie zmieniając struktur zapytań.





<<<OPISAC WSZYSTKIE ZNACZNIKI @ >>>

@updatedAt, pole oznaczone w taki sposób będzie przechowywało czas, kiedy ostatnio została dokonana edycja danego rekordu. Proces ten jest całkowicie automatyczny.

Znacznik @relation opisuje relacje jakie zachodzą między tabelami. Ze rysnunku XXX wynika, że tabela Feture połączona jest z tabelą User polami Feature.userId = User.id – zachodzi relacja „jeden do jeden”.

Ostatnia kolumna – executions to typ który jest tablicą – zachodzi relacja jeden do wielu. Oznacza to tyle, że jeden obiekt Feature, może być powiązany z wieloma obiektami ExecutionResult.

* Moduł „integration”

Katalog „integration” przechowuje funkcje które w jakiś sposób korzystają z zewnętrznych systemów. Wysyłanie powiadomienia email było bardzo ważną częścią aplikacji, a realizacja tego wymagała skorzystać z zewnętrznego serwisu Gmail.

Jak działa – pojedyncza instancja

Pokazanie wiadomości eamail

### Biblioteka

biblioteka Feature udostępnia jedna klasę o nazwie Feature. Do implementacji biblioteki autor zdecydował się użyć obiektowego języka C#. Konstrukcja samej biblioteki jest prosta, a jej działanie zobrazowano na rysunku XXX.



Klasa Feature implementuje interfejs IFeature, przedstawiony na rysunku XXX



Do utworzenie instancji Klasy Feature zastosowano wzorzec projektowy factory method . Metoda ControlledBy() tworzy obiekt Klasy feature i zwraca Interfejs IFeature. Każda instancja klasy obsługuje pojedyńczą funkcjonalność poprzez wskazanie indetyfikatora UUID w parametrze metody ControlledBy().

Zastosowanie interfejsu spowodowało, że deweloper będzie się komunikować się z Klasa Feature poprzez metody udostępnione w interfejsie. Zasada enkapsulacji mówi, aby udostępniać na zewnątrz tylko to co niezbędne do poprawnego funkcjonowania obiektu. Ułatwia to programiście korzystanie z klasy. Prowadzi do ograniczenia niepożądanych zmian w kodzie klasy bazowej.

Metody interejsu były implentowane zgodnie z wzorcem chain method. W którym to łączymy wywoływania metod w swoisty łańcuch. Dzięki temu poprawimy czytelność kodu.

Programista stosując tę bibliotekę uniknie niepożądanych skutków wywoływanych przez błędów w aplikacji. Zaprojektowano ją w taki sposób, że działanie aplikacji w której zaimplementowano jej użycie nie przerwie się z powodu wystąpienia błędu w przekazanym kodzie.

Kod programu kryjący się pod parametrami funkcji Replace() oraz With(), przekazywany jest dalej do metody RunCode() i wykonywany w bloku try{} catch{} - ukazano to na rysunku XXX. W przypadku wystąpienia błędu, wyjątek zostanie złapany, a następnie wynik wykonania kodu (obiekt Execution) przekazany do serwisu.



<<<DIAGRAM SEKWENCJI>>>

## Testy

Testy Api

- GET – zły uuid

- POST na zły uuid jak zachowuje się platforma

- zła struktura execrestult

## Eksploatacja

Opisać bibliotekę

Diagram aktywności - jak bibioliteka działa

Opisać programik c#

Opisać zasosoanie bibioteki

# Możliwe ścieżki rozwoju aplikacji

Rozbudowanie o Roloout

Wprowadzenie możliwości definiowania wartości typu string, numeber, boolean – i dalczego byłoby to lepsze

Zaimplementowanie lepszego łapania błędów (nr błędu, opis) i wysyłanie ich do service co należałoby zrobić – dodać diagram bazy danych jak mogłobyto wygladać

Zrobić link do email – aby moc szybko zobaczyć błąd wysłany w email, i załączyć przycisk on/off na stronce

Przerobic clienta tak aby mogl bys kontrolowany za pomocą wartości true/false z plikow config

# Podsumowanie

Cos tu będzie ………………………………………………………………………..

# Literatura / Załączniki

Cos tu będzie ………………………………………………………………………..

# Streszczenie

bb Jakos to chyba dziala teraz.

Jakos to chyba dziala teraz.