

# Wprowadzenie do zarządzania projektami deweloperskimi

## laboratorium



Karpiński Maciej  
Krysa Marcin  
Kuczma Łukasz  
Mertuszka Adam

Prowadzący mgr inż. Marcin Tracz

## Spis treści

<b>1</b>	<b>Opis projektu</b>	<b>3</b>
<b>2</b>	<b>Role w grupie</b>	<b>4</b>
<b>3</b>	<b>Wymagania i założenia projektowe</b>	<b>5</b>
<b>4</b>	<b>Opis działania</b>	<b>6</b>
<b>5</b>	<b>Wykorzystana technologia i narzędzia</b>	<b>7</b>
5.1	ASP.NET Core 3.1 . . . . .	7
5.2	AutoMapper . . . . .	7
5.3	C# . . . . .	7
5.4	Coverlet . . . . .	7
5.5	CSS . . . . .	8
5.6	Docker . . . . .	8
5.7	Entity Framework . . . . .	8
5.8	Fluent Assertions . . . . .	8
5.9	HTML . . . . .	9
5.10	JavaScript . . . . .	9
5.11	MailKit . . . . .	9
5.12	Microsoft SQL Server . . . . .	10
5.13	Node.js . . . . .	10
5.14	Quartz.NET . . . . .	10
5.15	React.js . . . . .	10
5.16	SEQ . . . . .	10
5.17	Serilog . . . . .	11
5.18	Swashbuckle . . . . .	11
5.19	xUnit.net . . . . .	11
<b>6</b>	<b>Event storming</b>	<b>12</b>
<b>7</b>	<b>Backlog</b>	<b>15</b>
<b>8</b>	<b>Estymata</b>	<b>19</b>
<b>9</b>	<b>Sprinty</b>	<b>20</b>
<b>10</b>	<b>Interfejs aplikacji</b>	<b>24</b>
<b>11</b>	<b>Uruchomienie w środowisku developerskim</b>	<b>25</b>
<b>12</b>	<b>Podsumowanie</b>	<b>27</b>
<b>13</b>	<b>Spis tabel i rysunków</b>	<b>28</b>

## 1 Opis projektu

Projekt „Magazynek piwosza” jest aplikacją internetową przeznaczoną dla pasjonatów warzących piwo w domu, którzy chcieliby posiadać listę posiadanych składników. „Magazynek piwosza” pozwala zarejestrowanemu i zalogowanemu użytkownikowi na tworzenie swoich wirtualnych „schowków” i przypisywania do nich składników wraz z ich datami ważności. Funkcjonalność aplikacji obejmuje:

- Dostęp do aplikacji poprzez przeglądarkę www (PC, smartphone)
- rejestrację i logowanie użytkowników,
- dodawanie, modyfikowanie i usuwanie kategorii przedmiotów,
- dodawanie, modyfikowanie i usuwanie przedmiotów,
- dodawanie, modyfikowanie i usuwanie magazynów,
- wysyłanie powiadomień o zbliżającym się terminie ważności,

## 2 Role w grupie

Każdy z członków zespołu posiada następujące role:

- Karpiński Maciej:
  - Backend
  - Baza danych
  - DevOps
  - Dokumentacja
  - Testy integracyjne
- Krysa Marcin
  - Frontend
- Kuczma Łukasz
  - Backend
- Mertuszka Adam
  - Frontend
  - Lider
  - Project Manager

### **3 Wymagania i założenia projektowe**

Wymagania projektowe:

- Uwierzytelnianie (w tym poprzez social media);
- Przechowywanie różnych typów produktów;
- Powiadomienia;
- Poręczny interfejs;
- Separacja backendu i frontendu;
- Logi.

Założenia:

- Użytkownik może tworzyć swoje własne schowki;
- Użytkownik wybiera składnik z listy składników i dodaje go do swojego schowka;
- Użytkownik wprowadza termin ważności składnika;
- Użytkownik dostaje powiadomienie o zbliżającym się terminie ważności;
- Przyjazny interfejs użytkownika;

## 4 Opis działania

Aplikacja pierwotnie stworzona dla miłośników piwowarstwa, ma za zadanie pomóc w zarządzaniu składnikami w swoich magazynkach, które niezbędne są do warzenia piwa. Wszystkie dodane magazyny są indywidualne i spersonalizowane, stąd też program wymaga od użytkownika zalogowania się albo poprzez rejestrację na stronie, albo używając mediów społecznościowych (gmail, facebook). Aby użytkownik dodał swoje produkty, musi na samym początku dodać swój pierwszy magazyn. Składów może być wiele - użytkownik może je podzielić wedle uznania kategoriami lub użyć jednego magazynu, aby trzymać w nim wszystkie składniki.

Każdy produkt posiada swoją indywidualną datę przydatności do spożycia, dlatego też przy dodawaniu przedmiotu do magazynu, niezbędne jest podanie daty przydatności do spożycia.

Aplikacja raz dziennie zbiera wszystkie informacje na temat produktów dodanych w magazynach, sprawdza daty przydatności i wysyła spersonalizowane powiadomienia mailowe z informacją, który składnik przeterminuje się lub który już jest przeterminowany.

Gotowa aplikacja przystosowana będzie typowo pod miłośników piwowarstwa, więc użytkownicy, przy dodawaniu przedmiotów do swoich magazynów, będą mogli wybierać predefiniowane składniki tylko z kategorii „Piwowarstwa”, jednak aplikacja została stworzona w taki sposób, aby umożliwić przyszłościowo rozbudowanie bibliotek produktowych o np. „nabiał”, czy „produkty mięsne”.

## 5 Wykorzystana technologia i narzędzia

Przy tworzeniu projektu aplikacji wykorzystano następujące technologie:

### 5.1 ASP.NET Core 3.1

ASP.Net Core jest wysokowydajnym frameworkm, do budowania nowoczesnych aplikacji internetowych wykorzystujących moc obliczeniową chmur. ASP.Net Core jest technologią open - source, wykorzystującą silnik html Razor, dzięki której możliwe jest tworzenie aplikacji multiplatformowych, które mogą być używane na każdym urządzeniu wyposażonym w przeglądarkę internetową.

### 5.2 AutoMapper

AutoMapper jest biblioteką służącą do mapowania między obiektami, dzięki czemu można automatycznie mapować właściwości dwóch różnych obiektów, przekształcając obiekt wejściowy jednego typu na obiekt wyjściowy innego typu.

### 5.3 C#

C# jest obiektowym językiem programowania, zaprojektowanym w latach 1998 – 2001 dla firmy Microsoft. Napisany program jest komplikowany do Common Intermediate Language(CLI), który następnie wykonywany jest w środowisku uruchomieniowym takim jak .NET Framework, .NET Core, Mono lub DotGNU. Wykorzystanie CLI sprawia, że kod programu jest wieloplatformowy (dopóki istnieje odpowiednie środowisko uruchomieniowe). C# posiada wiele wspólnych cech z językami Object Pascal, Delphi, C++ i Java a najważniejszymi cechami C# są:

- Obiektywość z hierarchią o jednym elemencie nadzewnętrznym (podobnie jak w Javie);
- Zarządzaniem pamięcią zajmuje się środowisko uruchomieniowe;
- Właściwości i indeksery;
- Delegaty i zdarzenia – rozwinięcie wskaźników C++;
- Typy ogólne, generyczne, częściowe, Nullable, domniemane, anonimowe;
- Dynamiczne tworzenie kodu;
- Metody anonimowe;
- Wyrażenia lambda.

### 5.4 Coverlet

Coverlet to projekt typu open source, który zapewnia wieloplatformowy framework pokrywający kod. Coverlet zbiera dane dotyczące przebiegu testu pokrycia, które są używane do generowania raportów.

## 5.5 CSS

Kaskadowe arkusze stylów (ang. Cascading Style Sheets, w skrócie CSS) – język służący do opisu formy prezentacji (wyświetlania) stron WWW. CSS został opracowany przez organizację W3C w 1996 r. jako potomek języka DSSSL przeznaczony do używania w połączeniu z SGML-em. Arkusz stylów CSS to lista dyrektyw (tzw. reguł) ustalających w jaki sposób ma zostać wyświetlana przez przeglądarkę internetową zawartość wybranego elementu (lub elementów) (X)HTML lub XML. Można w ten sposób opisać wszystkie pojedyncie odpowiedzialne za prezentację elementów dokumentów internetowych, takie jak rodzina czcionek, kolor tekstu, marginesy, odstęp międzywierszowy lub nawet pozycja danego elementu względem innych elementów bądź okna przeglądarki. Wykorzystanie arkuszy stylów daje znacznie większe możliwości pozycjonowania elementów na stronie, niż oferuje sam (X)HTML. CSS został stworzony w celu odseparowania struktury dokumentu od formy jego prezentacji. Separacja ta zwiększa zakres dostępności witryny, zmniejsza zawiłość dokumentu, ułatwia wprowadzanie zmian w strukturze dokumentu. CSS ułatwia także zmiany w renderowaniu strony w zależności od obsługiwanej medium (ekran, palmtop, dokument w druku, czytnik ekranowy). Stosowanie zewnętrznych arkuszy CSS daje możliwość zmiany wyglądu wielu stron naraz bez ingerowania w sam kod (X)HTML, ponieważ arkusze mogą być wspólne dla wielu dokumentów.

## 5.6 Docker

Docker jest otwarto źródłowym oprogramowaniem służącym do realizacji „konteneryzacji” aplikacji, służąca jako platforma dla programistów i administratorów do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych. Pozwala umieścić program oraz jego zależności (biblioteki, pliki konfiguracyjne, lokalne bazy danych itp.) w lekkim, przenośnym, wirtualnym kontenerze, który można uruchomić na prawie każdym serwerze z systemem Linux. Kontenery wraz z zawartością działają niezależnie od siebie i nie wiedzą o swoim istnieniu. Mogą się jednak ze sobą komunikować w ramach ściśle zdefiniowanych kanałów wymiany informacji. Dzięki uruchamianiu na jednym wspólnym systemie operacyjnym, konteneryzacja jest lżejszym sposobem wirtualizacji niż pełna wirtualizacja lub parawirtualizacja za pomocą wirtualnych systemów operacyjnych.

## 5.7 Entity Framework

Entity Framework jest technologią open - source do mapowania obiektowo – relacyjnego (ORM), które wspierają rozwój aplikacji zorientowanych na dane. Entity Framework umożliwia programistom pracę z danymi w postaci obiektów i właściwości specyficznych dla domeny, bez konieczności przejmowania się bazowymi tabelami i kolumnami baz danych, w których dane są przechowywane.

## 5.8 Fluent Assertions

Fluent Assertions to zestaw metod rozszerzających .NET, które pozwalają w bardziej naturalny sposób określić oczekiwany wynik testu jednostkowego. Umożliwia to prostą, intuicyjną budowę testu oraz szybsze diagnozowanie przyczyn niepowodzenia testu dzięki czytelniejszym błędom.

## 5.9 HTML

HTML (ang. HyperText Markup Language) – hipertekstowy język znaczników, wykorzystywany do tworzenia dokumentów hipertekstowych. HTML pozwala opisać strukturę informacji zawartych wewnątrz strony internetowej, nadając odpowiednie znaczenie semantyczne poszczególnym fragmentom tekstu – formując hiperłącza, akapity, nagłówki, listy – oraz osadza w tekście dokumentu obiekty plikowe np. multimedia bądź elementy baz danych np. interaktywne formularze danych. HTML umożliwia określenie wyglądu dokumentu w przeglądarce internetowej. Do szczegółowego opisu formatowania akapitów, nagłówków, użytych czcionek i kolorów, zalecane jest wykorzystywanie kaskadowych arkuszy stylów.

## 5.10 JavaScript

JavaScript (w skrócie JS) – skryptowy język programowania, stworzony przez firmę Netscape. Najczęściej spotykanym zastosowaniem języka JavaScript są strony internetowe. Skrypty te służą najczęściej do zapewnienia interakcji poprzez reagowanie na zdarzenia, walidacji danych wprowadzanych w formularzach lub tworzenia złożonych efektów wizualnych. Skrypty JavaScript uruchamiane przez strony internetowe mają znacznie ograniczony dostęp do komputera użytkownika. Po stronie serwera JavaScript może działać w postaci node.js lub Ringo. W języku JavaScript można także pisać pełnoprawne aplikacje. Fundacja Mozilla udostępnia środowisko złożone z technologii takich jak XUL, XBL, XPCOM oraz JSLib. Umożliwiają one tworzenie korzystających z zasobów systemowych aplikacji o graficznym interfejsie użytkownika dopasowującym się do danej platformy. Przykładem aplikacji napisanych z użyciem JS i XUL może być klient IRC o nazwie ChatZilla, domyślnie dołączony do pakietu Mozilla. Microsoft udostępnia biblioteki umożliwiające tworzenie aplikacji JScript jako część środowiska Windows Scripting Host. Ponadto JScript.NET jest jednym z podstawowych języków środowiska .NET. Istnieje także stworzone przez IBM środowisko SashXB dla systemu Linux, które umożliwia tworzenie w języku JavaScript aplikacji korzystających z GTK+, GNOME i OpenLDAP. Platforma Node.js umożliwia pisanie aplikacji wiersza poleceń oraz aplikacji serwerowych. Node.js używany jest także w środowisku Electron, który umożliwia pisanie aplikacji GUI. Język JavaScript używany jest także na urządzeniach internetu rzeczy, robotów czy układów takich jak Arduino poprzez bibliotekę Johnny-Five.

## 5.11 MailKit

MailKit jest multiplatformową otwarto źródłową biblioteką .NET klienta pocztowego opartą o MimeKit, która została zoptymalizowana pod kątem urządzeń mobilnych. MailKit oferuje następującą funkcjonalność:

- Obsługa proxy HTTP, Socks4, Socks4a i Socks5;
- Uwierzytelnianie SASL;
- Kompletny klient SMTP;
- Kompletny klient POP3;
- Kompletny klient IMAP;
- Sortowanie i wątkowanie wiadomości po stronie klienta;

- Asynchroniczne wersje wszystkich metod sieciowych;
- Obsługa S/MIME, OpenPGP, DKIM i ARC;
- Obsługa Microsoft TNEF.

## 5.12 Microsoft SQL Server

Microsoft SQL Server jest systemem zarządzania relacyjnymi bazami danych opracowany przez firmę Microsoft. Cechą charakterystyczną jest głównie wykorzystywanie języka zapytań Transact-SQL, który jest rozwinięciem standardu ANSI/ISO. W projekcie wykorzystano wersję 2019 Express, która jest bezpłatną edycją programu Microsoft SQL Server, oferującą podstawowy silnik bazy danych, nieposiadający ograniczenia ilości obsługiwanych baz lub użytkowników. Ograniczenia, występujące w wersji Express to m.in.: korzystanie z jednego procesora, 1 GB pamięci RAM, 10GB plików bazy danych czy brak SQL Agent.

## 5.13 Node.js

Node.js – wieloplatformowe środowisko uruchomieniowe o otwartym kodzie do tworzenia aplikacji typu server-side napisanych w języku JavaScript. Przyczynił się do stworzenia paradygmatu „JavaScript everywhere” umożliwiając programistom tworzenie aplikacji w obrębie jednego języka programowania zamiast polegania na odrębnych po stronie serwerowej. Node.js składa się z silnika V8 (stworzonego przez Google), biblioteki libUV oraz kilku innych bibliotek. Domyślnym managerem pakietów dla Node.js jest Npm.

## 5.14 Quartz.NET

Quartz.NET jest otwarto źródłową biblioteką do planowania zadań. Quartz.NET może być używany do tworzenia prostych lub złożonych harmonogramów wykonywania dziesiątek, setek, a nawet dziesiątek tysięcy zadań. Quartz.NET jest portem biblioteki Quartz dla środowiska Java.

## 5.15 React.js

React.js – biblioteka języka programowania JavaScript, która wykorzystywana jest do tworzenia interfejsów graficznych aplikacji internetowych. Często wykorzystywana do tworzenia aplikacji typu Single Page Application. Z głównych cech wyróżniających bibliotekę React.js jest wirtualny DOM (Document Object Model). React przechowuje cały DOM aplikacji w pamięci, po zmianie stanu wyszukuje różnice między wirtualnym i prawdziwym DOM i aktualizuje zmiany. Drugą z cech szczególnych React jest język JSX. Jest on nakładką na JavaScript, która dodaje możliwość wstawiania kodu html (lub komponentów React) bezpośrednio w kodzie, zamiast ciągu znaków. React.js jest obecnie używany na stronach internetowych firm takich jak Netflix, Imgur, PayPal, Archive.org, Gamepedia, SeatGeek, HelloSign czy Walmart.

## 5.16 SEQ

Seq jest zbudowany na potrzeby nowoczesnego, strukturalnego rejestrówania za pomocą szablonów wiadomości. Zamiast tracić czas i wysiłek na próbę wyodrębnienia danych

z dzienników w postaci zwykłego tekstu za pomocą delikatnego analizowania dziennika, właściwości skojarzone z każdym zdarzeniem dziennika są przechwytywane i wysyłane do Seq w czystym formacie JSON. Szablony wiadomości są obsługiwane natywnie przez ASP.NET Core, Serilog, NLog i wiele innych bibliotek.

## 5.17 Serilog

Serilog jest łatwą w konfiguracji biblioteką .NET zapewniającą podstawowe logowanie diagnostyczne do plików, konsoli itd. W przeciwieństwie do innych bibliotek rejestrowania dla .NET, parametry przekazywane wraz z komunikatami dziennika nie są destruktywnie renderowane do formatu tekstowego. Zamiast tego są zachowywane jako dane strukturalne, które można zapisać w formie dokumentu w magazynie danych NoSQL. Szablony komunikatów Serilog używają prostego DSL, który rozszerza zwykłe ciągi formatu .NET. Właściwości są nazywane w szablonie wiadomości i dopasowywane pozycyjnie z argumentami dostarczonymi do metody dziennika.

## 5.18 Swashbuckle

Swashbuckle jest biblioteką, która dodaje zestaw narzędzi „Swagger” generujących automatycznie dokumentację API aplikacji, wyposażoną w przejrzysty interfejs użytkownika. Swashbuckle umożliwia również testowanie API. Dokumentacja jest dostępna pod adresem internetowym:

<http://homebrew.tplinkdns.com:81/swagger>

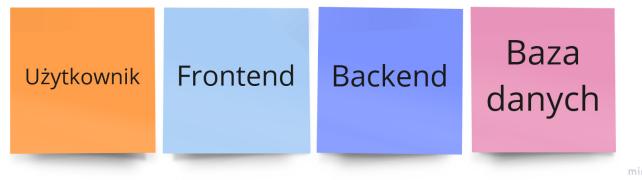
## 5.19 xUnit.net

xUnit.net to darmowe narzędzie typu open source służące do testowania jednostkowego przeznaczone dla platformy .NET Framework, napisane przez oryginalnego autora NUnit. xUnit.net współpracuje z platformami Xamarin, ReSharper, CodeRush i TestDriven.NET.

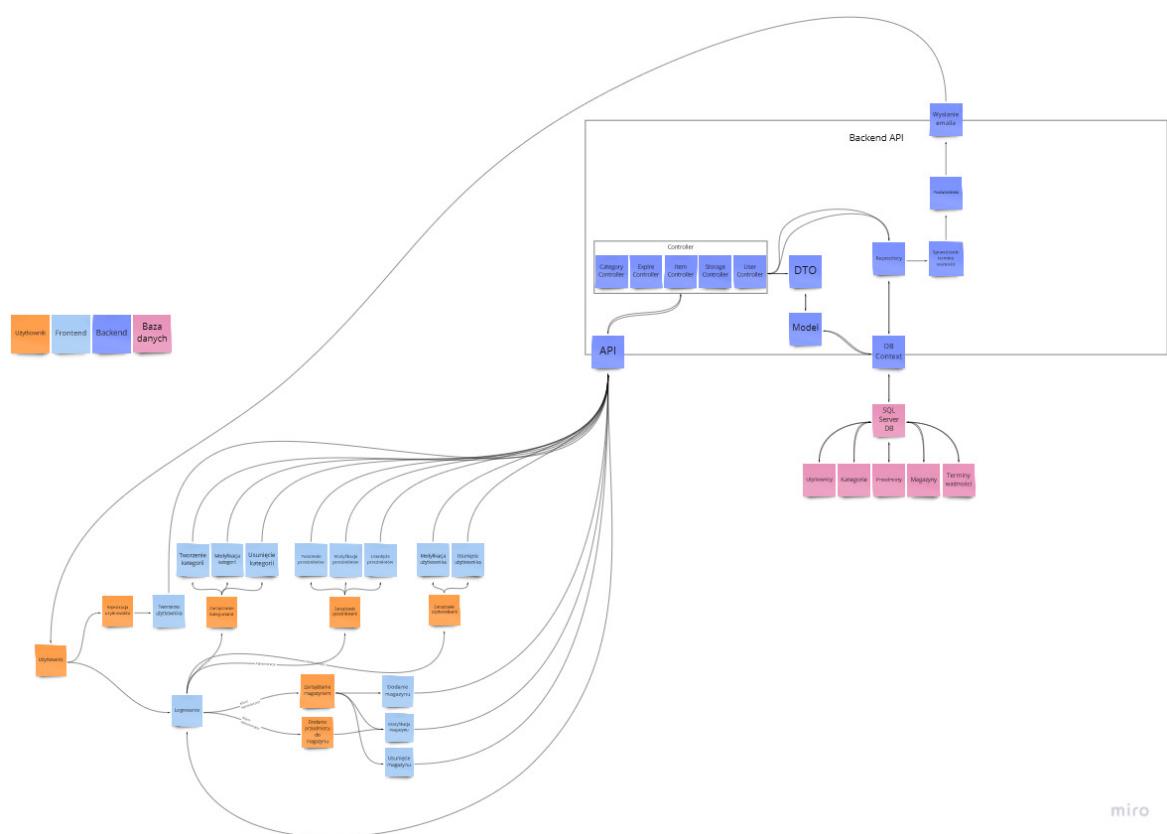
## 6 Event storming

Event Storming został przeprowadzony na portalu „Miro” <https://miro.com/>, w dniu 13.12.2020 było to nowe i ciekawe doświadczenie dla każdego z członków zespołu. Projekt w momencie przeprowadzenia event stormingu był już w zaawansowanej formie, wraz z rozpisanyimi zadaniami więc wykorzystaliśmy spotkanie do stworzenia modelu pracy aplikacji.

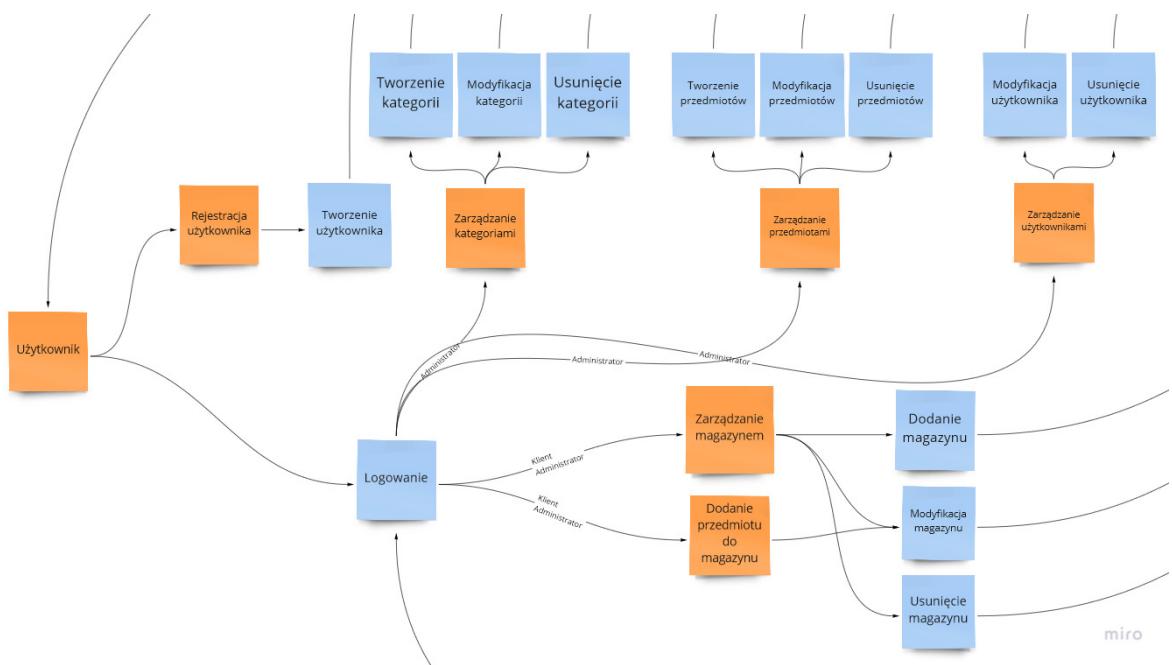
Podczas tworzenia wizualizacji okazało się, że rozumienie aspektów działania i reagowania aplikacji na niektóre zdarzenia różnią się między poszczególnymi członkami zespołu, ujawnione różnice w wyobrażeniu na temat funkcjonowania poszczególnych komponentów aplikacji spowodowały dyskusje nad wyborem najlepszego rozwiązania. Podczas spotkania wyraźnie było widać brak doświadczenia w tego typu zadaniu, na początku często przerywaliśmy sobie wypowiedzi ale dość szybko doszliśmy do porozumienia i zapanował porządek. W miarę rozwoju aplikacji do modelu były dodawane kolejne szczegóły. Wynikiem event stormingu jest spójny model działania aplikacji, dzięki czemu każdy członek zespołu może prześledzić przepływ informacji.



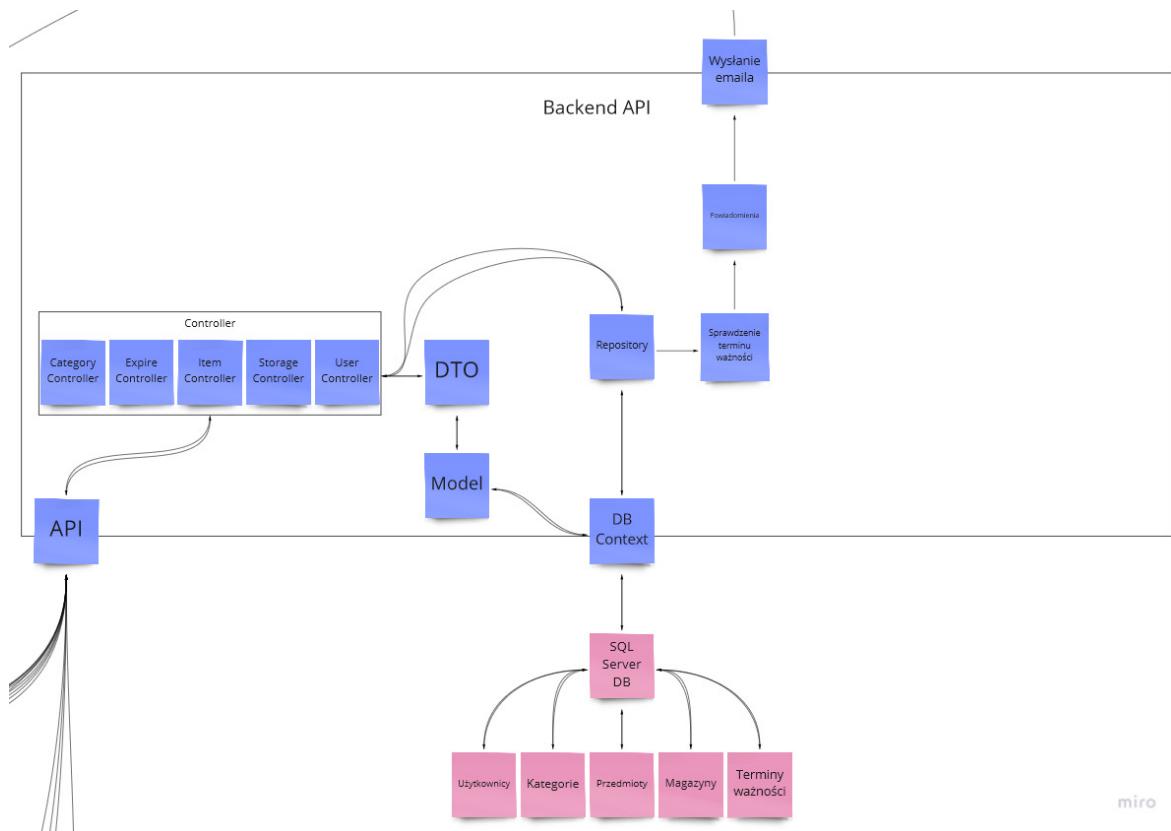
Rysunek 1: Miro - Legenda



Rysunek 2: Miro - Schemat działania aplikacji



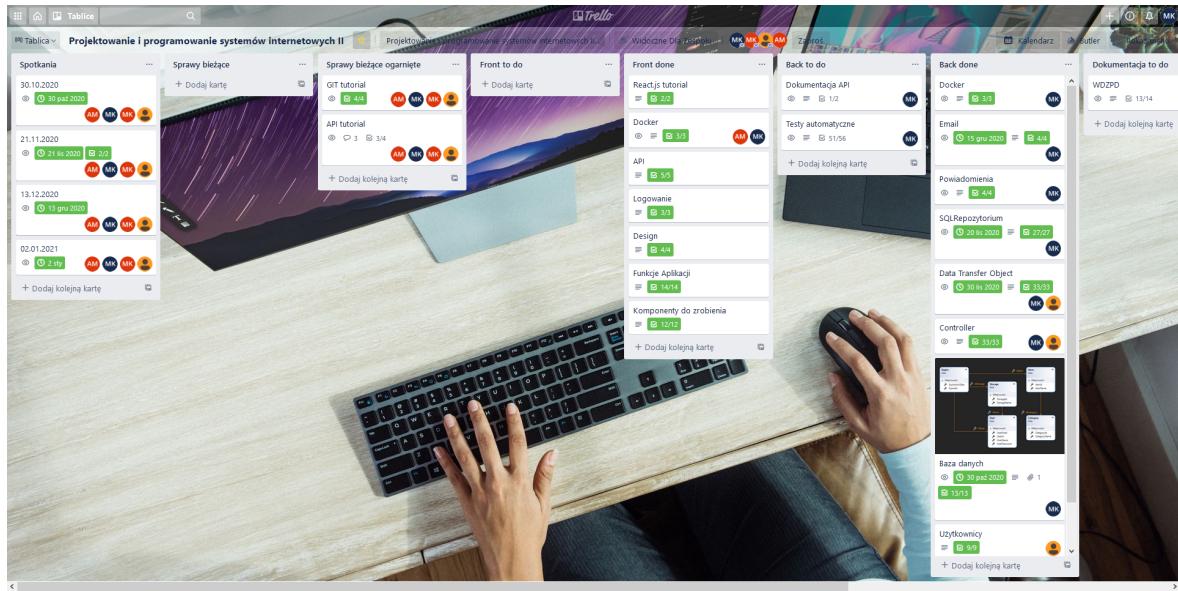
Rysunek 3: Miro - Frontend aplikacji



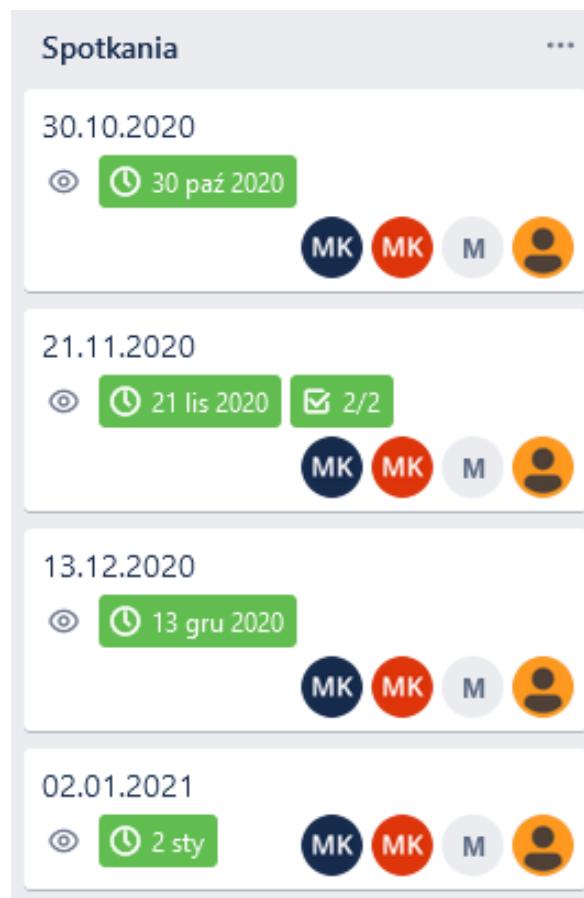
Rysunek 4: Miro - Backend aplikacji

## 7 Backlog

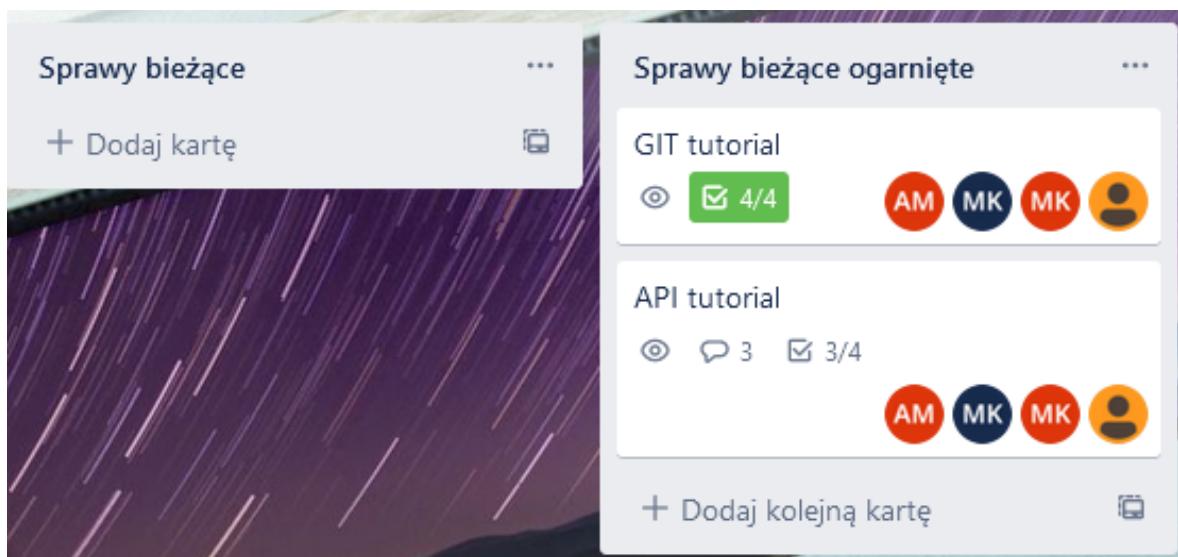
Backlog zaplanowanych działań został rozpisany na platformie „Trello”



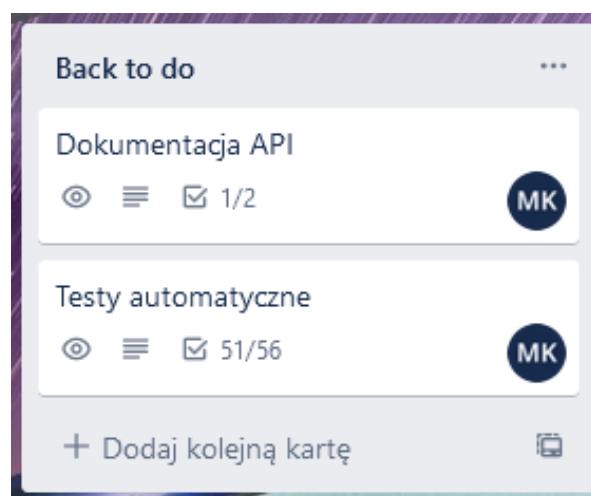
Rysunek 5: Backlog na platformie Trello



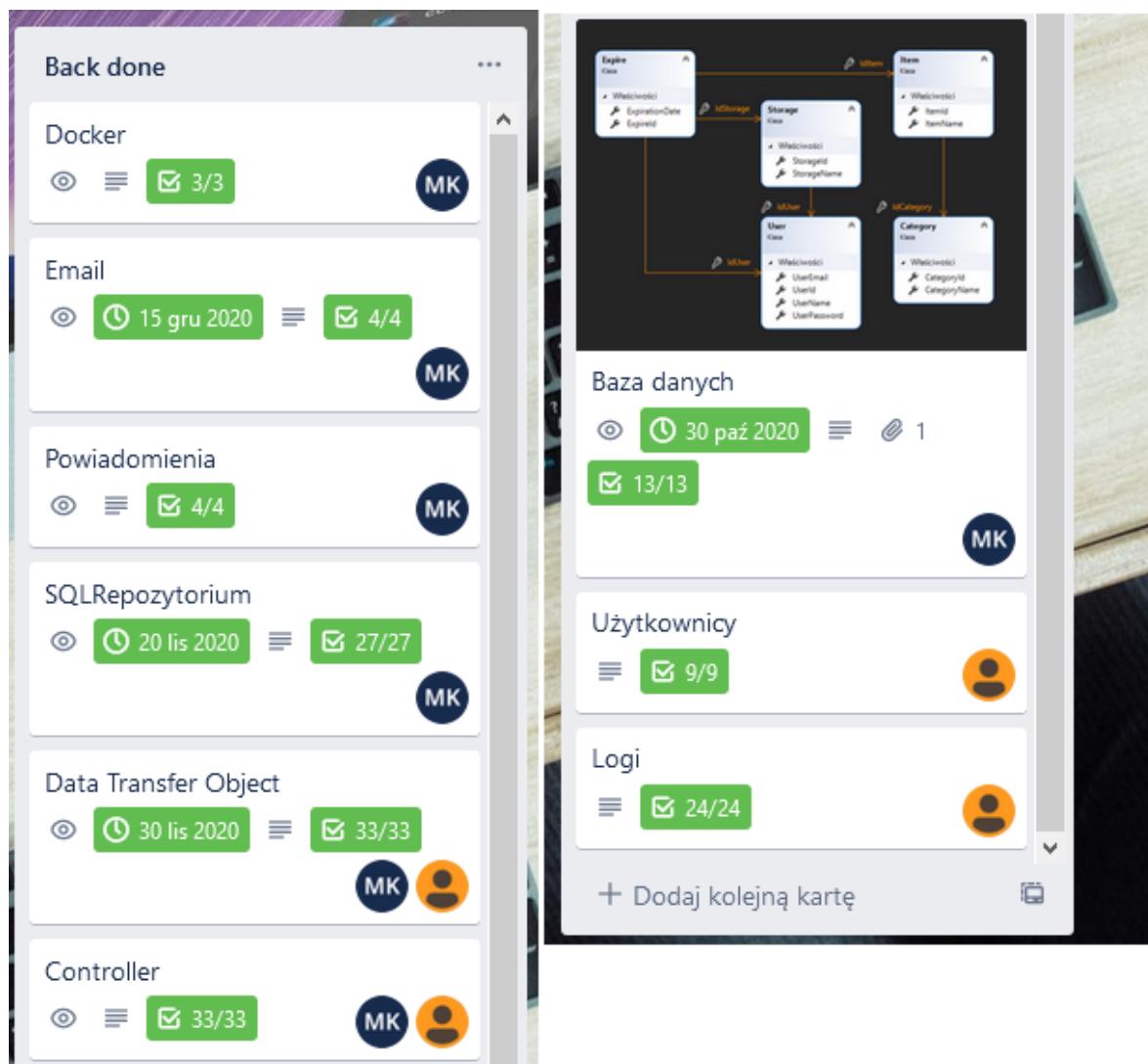
Rysunek 6: Terminy spotkań



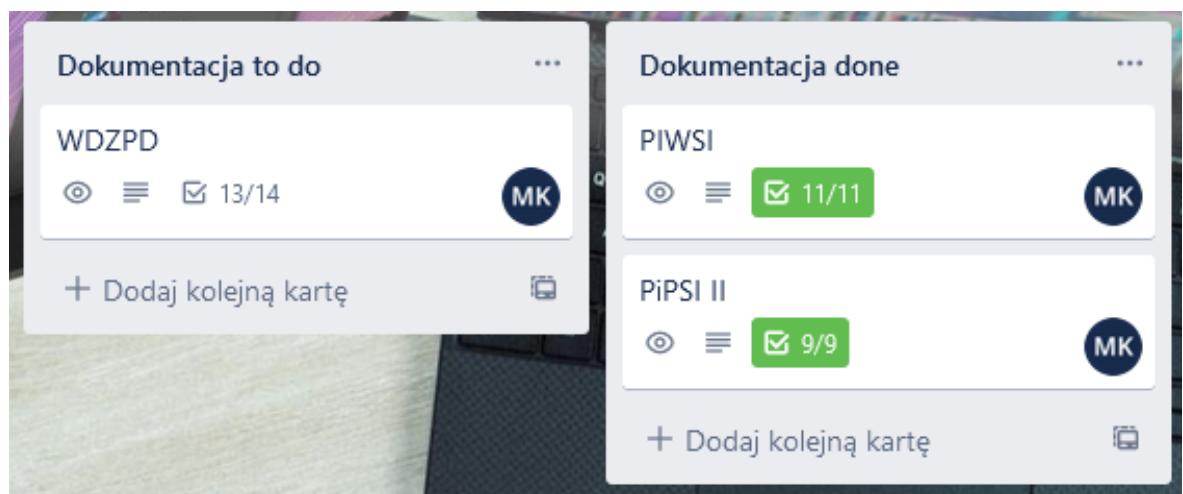
Rysunek 7: Sprawy bieżące



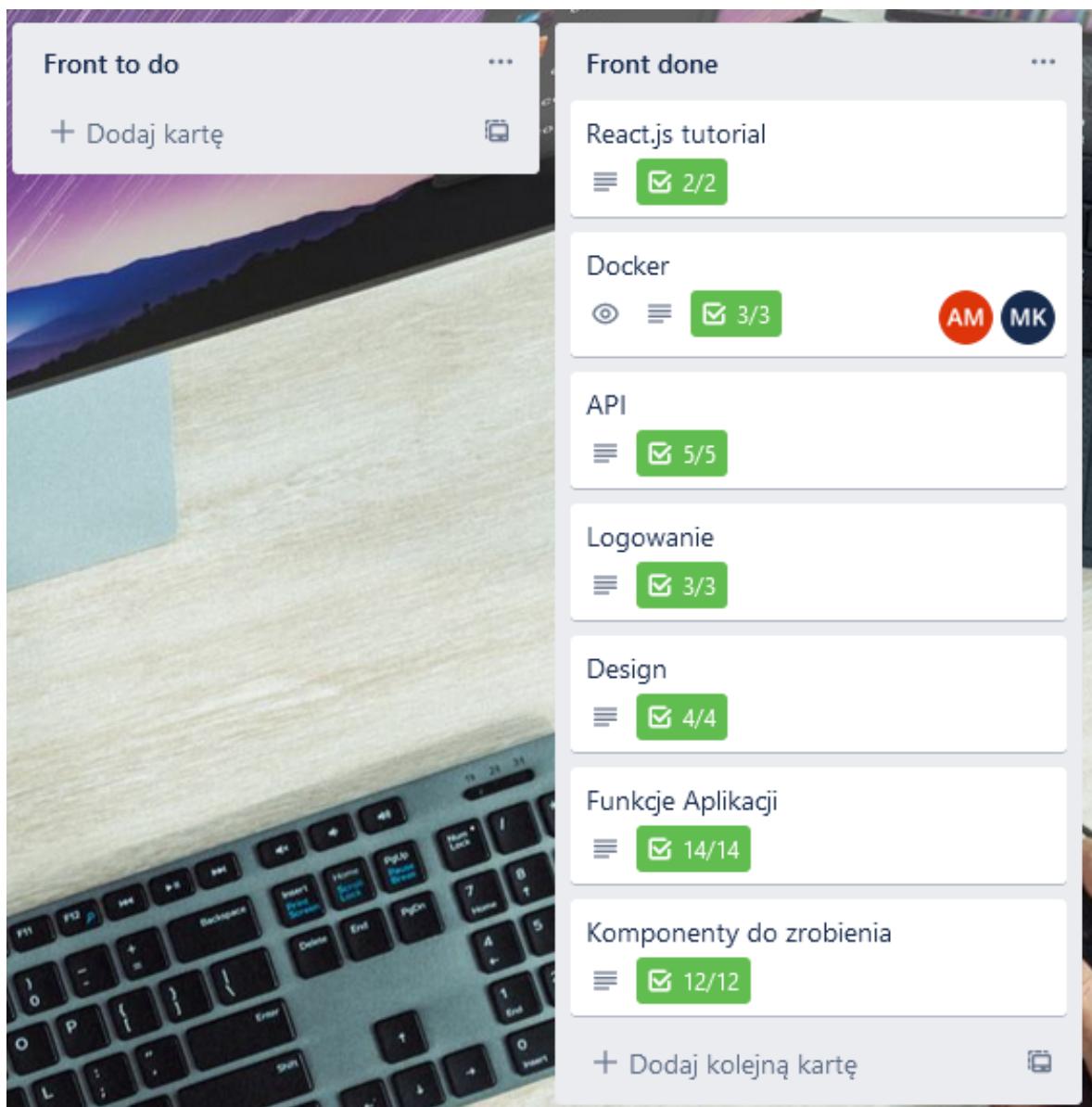
Rysunek 8: Backend - TODO



Rysunek 9: Backend - Zrealizowane



Rysunek 10: Dokumentacja



Rysunek 11: Frontend

## 8 Estymata

Szacowana estymacja poszczególnych części realizacji projektu:

- Backend:
  - Autoryzacja: 10 godzin,
  - Data Transfer Object (DTO): 2 godziny,
  - DevOps: 5 godzin,
  - End Point: 15 godzin,
  - Harmonogram: 3 godziny,
  - Logi: 10 godzin,
  - Programowanie logiki aplikacji: 50 godzin,
  - SQL Repozytorium: 3 godziny,
  - Testy integracyjne: 10 godzin,
  - Uwierzytelnianie: 10 godzin,
  - Wysyłanie e-maili: 30 minut,
- Baza danych:
  - Konfiguracja: 30 minut,
  - Projekt tabel: 2 godziny,
- Dokumentacja projektu: 5 godzin,
- Frontend:
  - Api: 10 godzin,
  - Logowanie: 2 godziny,
  - DevOps: 3 godziny,
  - Design Aplikacji: 5 godzin,
  - Konstruowanie aplikacji: 8 godzin,
  - Tworzenie komponentów: 10 godzin,
  - Pisanie funkcji: 15 godzin,
- Zarządzanie projektem:
  - Rozpisanie backlogu: 2 godziny,
  - Rozpisanie sprintów: 2 godziny,

## 9 Sprinty

Ze względu na różne harmonogramy pracy zawodowej poszczególnych członków zespołu, trudno było wyznaczyć szczegółowy termin realizacji pojedynczych komponentów aplikacji, więc zostały wyznaczone jedynie ostateczne terminy elementów, na których bazują inne części aplikacji. Pozostałe elementy były realizowane w drugiej kolejności.

Backend:

The screenshot shows a project management interface with a card for a database configuration. The card includes fields for 'CZŁONKOWIE' (MK), 'TERMIN' (30 paź 2020 o 23:59, status: ZREALIZOWANY), 'Opis' (Konfiguracja bazy danych), and 'Załączniki' (DB model.png). A sidebar on the right provides options for adding to a card (Członkowie, Etykiety, Lista zadań, Termin, Załącznik) and adding attachments (Dodaj dodatki Po...). Below the card is a list of tasks (Lista zadań) with checkboxes and a progress bar at 100%. The tasks listed include: connection string, initial migration, user-model, item-model, storage-model, expire-model, category-model, update-expire-model aby używał user id, update-expire-model aby używał poprawnego formatu daty, update-storage-model aby używał user id, update-migration po zmianach w modelach, and update-migration po dodaniu użytkownia. A 'DZIAŁANIA' sidebar on the right offers actions like Przenieś, Kopij, Utwórz szablon, Obserwuj, Zarchiwizuj, and Udostępnij.

Rysunek 12: Sprinty - Baza danych

**SQLRepozytorium**

na liście [Back done](#)

CZŁONKOWIE TERMIN

MK + 20 lis 2020 o 23:59 **ZREALIZOWANY**

**Opis** Edytuj

Repozytorium metod do obsługi bazy danych

**Lista zadań** 100% Ukryj ukończone elementy Usuń

SaveChanges  
 GetAllCategories  
 GetCategoryById  
 GetAllExpires  
 GetExpireById  
 GetAllItems  
 GetItemById  
 GetAllStorages  
 GetStorageById  
 CreateCategory  
 CreateExpire  
 CreateItem  
 CreateStorage  
 UpdateCategory  
 UpdateExpire  
 UpdateItem  
 UpdateStorage  
 DeleteCategory

**DODAJ DO KARTY**

Członkowie  
 Etykiety  
 Lista zadań  
 Termin  
 Załącznik  
 Okładka

**DODATKI**

+ Dodaj dodatki Po...  
Ulepsz, aby uzyskać nielimitowaną liczbę dodatków Power-Up na tablicę.  
 Ulepsz zespół

**BUTLER** **NOWE**   
+ Dodaj przycisk karty

**DZIAŁANIA**

Przenieś  
 Kopiuj  
 Utwórz szablon  
 Obserwuj   
 Zarchiwizuj  
 Udostępnij

Rysunek 13: Sprinty - Repozytorium SQL

### Data Transfer Object

na liście [Back done](#)

CZŁONKOWIE TERMIN

MK + 30 lis 2020 o 23:59 | ZREALIZOWANY

Opis Edytuj

Data transfer object - moduł pobiera dane z bazy danych (SqlBackendRepo), mapuje je(Profile) obrabia je (Dtos) i wysyła do controllera

Dtos

Ukryj ukończone elementy Usuń

100%

- CategoryReadDto
- CategoryCreateDto
- ExpireReadDto
- ExpireCreateDto
- ItemReadDto
- ItemCreateDto
- StorageReadDto
- StorageCreateDto
- ItemReadDto — zaktualizować mapowanie
- ExpireReadDto — zaktualizować mapowanie
- StorageReadDto — zaktualizować mapowanie
- ItemCreateDto — zaktualizować mapowanie
- ExpireCreateDto — zaktualizować mapowanie
- StorageCreateDto — zaktualizować mapowanie
- ItemUpdateDto — zaktualizować mapowanie
- ExpireUpdateDto — zaktualizować mapowanie
- StorageUpdateDto — zaktualizować mapowanie

DODAJ DO KARTY

Członkowie

Etykiety

Lista zadań

Termin

Załącznik

Okładka

DODATKI

+ Dodaj dodatki Po...

Ulepsz, aby uzyskać nielimitowaną liczbę dodatków Power-Up na tablicę.

BUTLER NOWE

+ Dodaj przycisk karty

DZIAŁANIA

→ Przenieś

✉ Kopiuj

⬇ Utwórz szablon

Obserwuj

✉ Zarchiwizuj

🔗 Udostępnij

Rysunek 14: Sprinty - Data Transfer Object

**Email**

na liście [Back done](#)

CZŁONKOWIE TERMIN

+ 15 gru 2020 o 23:59 **ZREALIZOWANY**

**Opis** Edytuj

Usługa wysyłania emaili

Dane do konta email:  
login: [storage.homebrew@gmail.com](#)  
hasło: [REDACTED]

**Lista zadań**

100%

- [Dedanie usługi email](#)
- [Dedanie docker-environment do docker-compose-backend i docker-compose-backend-dev](#)
- [Załeganie konta email](#)
- [Aktualizacja docker-compose-backend i docker-compose-backend-dev o dane serwera email](#)

[Dodaj element...](#)

[Ukryj ukończone elementy](#) [Usuń](#)

**DODAJ DO KARTY**

- Członkowie
- Etykiety
- Lista zadań
- Termin
- Załącznik
- Okładka

**DODATKI**

+ Dodaj dodatki Po...  
Ulepsz, aby uzyskać nielimitowaną liczbę dodatków Power-Up na tablicę.  
[Ulepsz zespół](#)

**BUTLER** **NOWE**   
+ Dodaj przycisk karty

Rysunek 15: Sprinty - usługa wysyłania e-maili

## 10 Interfejs aplikacji

## 11 Uruchomienie w środowisku developerskim

Repozytorium kodu projektu i jego dokumentacja znajdują się w serwisie „GitHub” pod adresem:

<https://github.com/MacKarp/Homebrewing-storage>

Do uruchomienia systemu wymagana jest działająca platforma konteneryzacji „Docker” oraz narzędzie „Docker Compose” szczegóły instalacji i konfiguracji wymaganych komponentów, można sprawdzić na stronie internetowej: <https://docs.docker.com/>.

Pościągnięciu repozytorium w katalogu głównym projektu znajdują się pliki wsadowe, umożliwiające komplikację i uruchomienie „frontendu” i „backendu” wraz z bazą danych. Pliki wsadowe dla systemu Windows:

- Backend-Start.bat
- Backend-Start-DEV.bat
- Backend-Stop.bat
- Backend-Stop-DEV.bat
- Frontend-Start.bat
- Frontend-Start-DEV.bat
- Frontend-Stop.bat
- Frontend-Stop-DEV.bat

Pliki dla systemu Linux:

- Backend-Start.sh
- Backend-Start-DEV.sh
- Backend-Stop.sh
- Backend-Stop-DEV.sh
- Frontend-Start.sh
- Frontend-Start-DEV.sh
- Frontend-Stop.sh
- Frontend-Stop-DEV.sh

Pliki „Backend-Start.bat”, „Backend-Stop.bat” oraz „Backend-Start.sh” i „Backend-Stop.sh” służą do uruchomiania i zatrzymywania „backendu” w wersji produkcyjnej, która pobierze i uruchomi najnowszą stabilną wersję „backendu” aplikacji i bazę danych z serwisu „Docker Hub”, w przypadku pliku z przyrostkiem „DEV” zostanie uruchomiona komplikacja wersji deweloperskiej znajdującej się w lokalnym katalogu „Backend” a następnie zostanie utworzony kontener z działającą aplikacją. W przypadku chęci zmiany ustawień „backendu” aplikacji lub kontenera należy edytować - zachowując odpowiednie formatowanie plików yaml - plik w wersji produkcyjnej lub deweloperskiej:

- docker-compose-backend.yaml
- docker-compose-backend-dev.yaml

Pliki „Frontend-Start.bat”, „Frontend-Stop.bat” oraz „Frontend-Start.sh” i „Frontend-Stop.sh” służą do uruchomiania i zatrzymywania „frontendu” aplikacji w wersji produkcyjnej, która pobierze i uruchomi najnowszą stabilną wersję „frontendu” z serwisu „Docker Hub”, w przypadku pliku z przyrostkiem „DEV” zostanie uruchomiona kompilacja wersji deweloperskiej znajdującej się w lokalnym katalogu „Frontend” a następnie zostanie utworzony kontener z działającą aplikacją. W przypadku chęci zmiany ustawień „frontendu” aplikacji lub kontenera należy edytować - zachowując odpowiednie formatowanie plików yaml - plik w wersji produkcyjnej lub deweloperskiej:

- docker-compose-frontend.yaml
- docker-compose-frontend-dev.yaml

Domyślna konfiguracja znajdująca się w repozytorium umożliwia osobne uruchomienie „frontendu” i „backendu” wraz z bazą danych na jednym komputerze. Poszczególne komponenty aplikacji są dostępne pod adresami:

- Backend: <http://localhost:8080>
- Frontend: <http://localhost>:
- Logi: <http://localhost:5341>

Domyślne dane dostępowe do bazy danych:

- Nazwa użytkownika: SA
- Hasło użytkownika: ThisIsNotSuperSecretP@55w0rd
- Adres: localhost
- Port: 14331

## 12 Podsumowanie

Sprawne zarządzanie projektem jest trudnym zadaniem i wymaga aby lider projektu był osobą charyzmatyczną.

Zarządzanie zespołem, w którym każdy z członków posiada różny harmonogram pracy zawodowej sprawiło wiele problemów organizacyjnych, często każdy z członków zespołu był dostępny w innym terminie, przez co ustalenie terminu spotkania online, w którym każdy z członków zespołu byłby dostępny było trudnym zadaniem, z tego powodu w czasie trwania projektu dominowała komunikacja pisana. Z tych samych względów rozpisywanie sprintów ograniczyło się jedynie do określenia nieprzekraczalnego terminu ukończenia elementów, od których inne komponenty projektowanej aplikacji były uzależnione, pozostałe elementy nie posiadały ściśle określonego terminu i były realizowane w drugiej kolejności.

Każdy z członków zespołu zdobył dużo wiedzy na temat pracy w zorganizowanym zespole, a także poznał narzędzia ułatwiające pracę w takim zespole.

Wspólnie doszliśmy do wniosku że Event Storming był nowym i ciekawym doświadczeniem, chodź lepszym byłoby spotkanie „twarzą w twarz” wraz z osobą doświadczoną, która mogłaby poprowadzić takie spotkanie.

Wszyscy członkowie zespołu zgodnie stwierdzili że największym problem podczas pracy był problem z bieżącą komunikacją oraz z początkowym brakiem zrozumienia działania tablic na platformie „Trello”, raz zdarzyło się nawet że dwie osoby zrobiły tą samą część aplikacji - na szczęście strata czasu w tym wypadku była mała.

## 13 Spis rysunków

### Spis rysunków

1	Miro - Legenda . . . . .	12
2	Miro - Schemat działania aplikacji . . . . .	13
3	Miro - Frontend aplikacji . . . . .	13
4	Miro - Backend aplikacji . . . . .	14
5	Backlog na platformie Trello . . . . .	15
6	Terminy spotkań . . . . .	15
7	Sprawy bieżące . . . . .	16
8	Backend - TODO . . . . .	16
9	Backend - Zrealizowane . . . . .	17
10	Dokumentacja . . . . .	17
11	Frontend . . . . .	18
12	Sprinty - Baza danych . . . . .	20
13	Sprinty - Repozytorium SQL . . . . .	21
14	Sprinty - Data Transfer Object . . . . .	22
15	Sprinty - usługa wysyłania e-maili . . . . .	23