**Projekt i implementacja systemu wspomagania zarządzaniem firmą wykonującą zakupy „Pandronka”**

**Autorzy i numery indeksu:**

Michał Baca 8316

Rafał Sawicki 8647

Mateusz Rzewnicki 8138

**Grupa:** Z712

Spis treści

[Analiza stanu zastanego 4](#_Toc61183917)

[Uber Eats – Mateusz Rzewnnicki 4](#_Toc61183918)

[Glovo – Michał Baca 5](#_Toc61183919)

[ABER SKLEP – Rafał Sawicki 7](#_Toc61183920)

[Podsumowanie 8](#_Toc61183921)

[Analiza systemu 9](#_Toc61183922)

[Koncepcja systemu - Mateusz Rzewnnicki 9](#_Toc61183923)

[Specyfikacja funkcjonalna 11](#_Toc61183924)

[Modele danych – Michał Baca 17](#_Toc61183925)

[Model architektury systemu – Rafał Sawicki 18](#_Toc61183926)

[Projekt architektury systemu 20](#_Toc61183927)

[1.1 Diagramy klas – Michał Baca 20](#_Toc61183928)

[1.2 Schemat warstw – Mateusz Rzewnicki 25](#_Toc61183929)

[1.3 Schematy modułów – Mateusz Rzewnicki 26](#_Toc61183930)

[Moduł Produktu 26](#_Toc61183931)

[Moduł Kuriera 26](#_Toc61183932)

[Moduły Producentów, Kategorii i Jednostki 27](#_Toc61183933)

[Moduł Zamówienia 27](#_Toc61183934)

[Moduł Koszyka 28](#_Toc61183935)

[Moduł Administratora 28](#_Toc61183936)

[Moduł Kont 29](#_Toc61183937)

[Projekt bazy danych – Michał Baca 30](#_Toc61183938)

[Projekt algorytmów – Mateusz Rzewnicki 36](#_Toc61183939)

[1.4 Diagram sekwencji logowania i rejestracji – Michał Baca 36](#_Toc61183940)

[1.5 Diagram sekwencji obsługi koszyka – Michał Baca 37](#_Toc61183941)

[1.7 Diagram sekwencji realizacji zamówienia przez kuriera – Mateusz Rzewnicki 39](#_Toc61183942)

[1.8 Diagram sekwencji wprowadzenia kuriera – Mateusz Rzewnicki 39](#_Toc61183943)

[Projekt interfejsu użytkownika - Rafał Sawicki 41](#_Toc61183944)

[1.9 Homepage 42](#_Toc61183945)

[1.10 Panel użytkownika 42](#_Toc61183946)

[1.11 Nowy produkt 44](#_Toc61183947)

[1.12 Koszyk 45](#_Toc61183948)

[1.13 Informacje o produkcie 46](#_Toc61183949)

[1.14 Rejestracja 46](#_Toc61183950)

[Kurier 48](#_Toc61183951)

[1.15 Produkty 49](#_Toc61183952)

[1.16 Płatność kartą 50](#_Toc61183953)

[Projekt testów kontrolnych funkcjonalne - Rafał Sawicki 50](#_Toc61183954)

Analiza stanu zastanego

### Uber Eats – Mateusz Rzewnnicki

Uber Eats to system stworzony do odnajdywania punktów żywieniowych w okolicy wskazanego adresu, działa on pod adresem [www.ubereats.com](http://www.ubereats.com/) . Po przekazaniu adresu system analizuje lokalizacje i proponuje nam wszystkie dostępne okoliczne restauracje oraz przy końcu listy te dalsze.

Ponadto zostały zaimplementowane w niej zaawansowane filtry, dzięki którym możemy precyzyjnie wskazać jakiego miejsca lub dania poszukujemy. Poza okolicznymi restauracjami wyszukiwać możemy między innymi piekarnie, z których możemy zamówić oferowane tam produkty nawet spożywcze, jeśli takowe oferuję. Aplikacja ma bardzo prosty i przejrzysty interfejs, który poprawnie skaluje się na wszystkich urządzeniach. Witryna jest czytelna, ale i nowoczesna, a oko umilają liczne animacje i płynne przejścia między komponentami.

Do zalet aplikacji zaliczyć możemy również dużą bazę punktów, które tylko dowodzą jej popularności, dzięki czemu oferta jest bardzo bogata. Jako kolejną zaletę aplikacji zauważyć trzeba integracje z popularną usługą Uber, dzięki czemu łatwo i szybko otrzymamy złożone zamówienie.

Do wad aplikacji należy ograniczona liczba produktów spożywczych jedynie do piekarni w ofercie, co sprawia, że w większości zamówić można jedynie gotowe potrawy, a nie produkty do ich stworzenia. Niestety do złożenia zamówienia wymagane jest autoryzowane konto w serwisie a skomplikowany proces jego utworzenia i potwierdzenia wydłuża znacząco całą aktywność dla głodnych klientów.

### Glovo – Michał Baca

Aplikacja Glovo jest to system pozwalający na zlecenie zakupu i dostawy produktów. W swojej ofercie firma posiada przekąski, jedzenie, artykuły spożywcze, apteki oraz inne sklepy. Do zlecenia transakcji wymagane jest konto na portalu.

System znajduję się pod adresem glovoapp.com. System na początek analizuje naszą lokalizację. Następnie możemy wybrać kategorię, co chcemy zlecić lub przejść do proponowanych i rekomendowanych ofert. Program oferuje nam pobranie aplikacji mobilnej co pozwoli na śledzenie dostawy.

Strona internetowa przyjmuje jasne barwy, przeważa żółty kolor z elementami koloru białego oraz zielonego. Aplikacja jest przejrzysta i responsywna.

System posiada takie funkcjonalności jak wyszukiwarka, rejestracja i logowanie, przeglądanie ofert po kategoriach. Jest możliwość zmiany lokalizacji. Po przeniesieniu się do widoku kategorii mamy możliwość sortowania bardziej szczegółowego. W prawym dolnym rogu można dostrzec przycisk, który przenosi nas na górę strony.



Do zalet aplikacji należy bez wątpienia przejrzysty wygląd i zachowany system logiki na kolejnych podstronach programu.

Do wad programu możemy zaliczyć żółty kolor, który jest przytłaczający. W aplikacji brakuje również sortowania po cenie produktu.

Niestety nie ma możliwości sprawdzenia technologii wykonania tego programu, co uniemożliwia analizę aplikacji pod względem technicznym i ocenę programu można uzasadnić tylko na podstawie udostępnionej aplikacji.

### ABER SKLEP – Rafał Sawicki



[**abersklep.pl**](https://abersklep.pl/)to supermarket internetowy zaprojektowany dla wygodnego zakupu produktów online wraz z dowozem. Na pierwszy rzut oka przesłanie serwisu jest jasne zapewnić użytkownikom szybkie i wygodne zakupy z tysiącami najwyższej jakości produktów.

Serwis zapewnia wiele niezbędnych funkcjonalności takich jak: rejestracja, logowanie, edytowalna lista zakupów, koszyk, wiele form płatności, szukajki, sortowania, filtrowania, zapis do newsletteru.

Główna zaletą serwisu jest intuicyjny interfejs oraz płynne przechodzenie między zakładkami, szeroki wybór kategorii oraz samych produktów a dodatkowo promocje na konserwy.   
Zawuażalne sądwie wady serwisu jjednato jest brak optymalizacji dla przeglądarki IE 9 i nizszych. A druga to słaba optymalizazcja aplikacji podwzględem SEO.

*Analiza techniczna:*

Czas pierwszego wczytywania: 3.57s

Czas ponownego wczytwania: 1.56s

Stosunek treści do kodu: 10%

Certyfikat SSL

Serwer Apache 2

Język skryptowy: PHP 7.2.33

**Wniosek: aplikacja udowadnia, że zakupy w sklepie online mogą być proste a właściciel jest gotowy na kompleksową realizację zamowienia.**

### Podsumowanie

W poniższej tabeli zostały zebrane wszystkie programu i poprzez oznaczenie literą „X” zaznaczono funkcjonalność, jeśli w danym systemie występuje. W przypadku występowania funkcji w każdym programie oznacza to, że funkcja ta jest pożądana.

|  |  |  |  |
| --- | --- | --- | --- |
| Funkcjonalność | UberEats | Glovo | ABER SKLEP |
| Sortowanie po kategorii | X | X | X |
| Przejrzysty wygląd | X | X | X |
| Obowiązkowa rejestracja | X | X | X |
| Responsywność | X | X | X |

Analiza systemu

### Koncepcja systemu - Mateusz Rzewnnicki

Celem pracy jest analiza w celu dalszego zaprojektowania systemu wspomagającego zarzadzaniem firmą wykonującą zakupy pod tytułem „Pandronka”. Ze względu na przejrzystość funkcje zostały podzielone na aktorów którzy z nich korzystają.



Poniżej przedstawiono role aktorów.

|  |  |
| --- | --- |
| **Aktor** | **Opis** |
| Administrator | Jest to pracownik firmy którzy zarządza serwisem wraz z jego zawartością oraz kontami użytkowników. |
| Użytkownik | Jest to klient który utworzył konto w serwisie. Posiada on dostęp do funkcji po uwierzytelnieniu. |
| Klient | Jest to potencjalny użytkownik który nie ma jeszcze konta w serwisie. Posiada on dostęp do podstawowych funkcji systemu. |
| Kurier | Jest to pracownik kompletujący i dostarczający zamówienie do zamawiającego. |

Głównym celem systemu jest umożliwienie Użytkownikom i Klientom zamówienie produktów żywnościowych lub innych drobniejszych towarów. Przewiduje się że system udostępniał będzie podstawowe funkcjonalności co do wyboru produktów i przeglądania oferty serwisu, złożenie zamówienia możliwe będzie dopiero po uwierzytelnieniu poprzez założenie konta. Założenie konta będzie rozszerzało dostępne funkcje o dodatkowe jak przeglądanie historii swoich zamówień lub planowanie przyszłych.

System umożliwiał będzie dokonanie płatności po przez portal PayU który udostępnia wiele form płatności lub płatność przy odbiorze za pomocą gotówki. Po utworzonym zamówieniu i płatności, lista zostanie przekazana do kuriera który skompletuje zamówienie i dostarczy zamawiającemu.

Do zadań Administratora należy dodawanie nowych kurierów oraz produktów do oferty na stronie. Ponadto może zarządzać kontami pracowników czyli Kurierami oraz kontami Użytkowników.

### Specyfikacja funkcjonalna

#### **Diagram hierarchii funkcji - Mateusz Rzewnnicki**

Poniżej znajduje się diagram głównych funkcji systemu przypisanych do aktorów, ich szczegółowy opis znajduje się w następnym podrozdziale

#### **Lista funkcji – Mateusz Rzewnicki**

W kilku poniższych tabelach znajduje się lista funkcji które powinny znaleźć się w systemie by pracownicy i klienci mogli realizować założenia biznesowe.

|  |  |
| --- | --- |
| **Identyfikator** | **Opis** |
| F\_1 | **Funkcje Administratora:**  F\_1.1 Użytkownicy  F\_1.1.1 Reset hasła  F\_1.1.2 Wyświetlenie listy użytkowników  F\_1.1.3 Podsumowanie użytkownika wraz ze wszystkimi jego zamówieniami i transakcjami  F\_1.2 Kurierzy  F\_1.2.1 Wprowadzenie imienia  F\_1.2.2 Wprowadzenie nazwiska  F\_1.2.3 Wprowadzenie loginu  F\_1.2.4 Wprowadzenie hasła  F\_1.2.5 Wprowadzenie adresu email  F\_1.2.6 Wprowadzenie numeru telefonu  F\_1.2.7 Wprowadzenie miast w których operuje  F\_1.2.8 Przydzielenie zlecenia  F\_1.2.9 Odebranie zlecenia  F\_1.2.10 Wyświetlenie listy obecnie realizowanych zleceń  F\_1.2.11 Podsumowanie wyników kuriera  F\_1.2.12 Reset hasła  F\_1.3 Produkty  F\_1.3.1 Wprowadzenie nazwy produktu  F\_1.3.2 Wprowadzenie ceny produktu  F\_1.3.3 Wprowadzenie unikalnej nazwy producenta  F\_1.3.4 Możliwość wprowadzenia marży produktu  F\_1.3.5 Wybór z predefiniowanych kategorii  F\_1.3.6 Kategorie  F\_1.3.6.1 Wprowadzenie nazwy kategorii  F\_1.3.6.2 Wprowadzenie marży kategorii  F\_1.4 Klienci  F\_1.4.1 Wyświetlenie listy klientów bez powiązanego konta użytkownika  F\_1.4.2 Wyświetlenie podsumowania klienta bez powiązanego konta wraz z wybranymi produktami |

|  |  |
| --- | --- |
| **Identyfikator** | **Opis** |
| F\_2 | **Funkcje Klienta:**  F\_2.1 Koszyk  F\_2.1.1 Dodanie produktu do koszyka  F\_2.1.2 Usunięcie produktu z koszyka  F\_2.1.3 Możliwość zmiany ilości konkretnego produktu  F\_2.1.4 Wybór metody płatności  F\_2.2 Wyszukiwanie  F\_2.2.1 Wyszukanie produktu przy użyciu jego nazwy  F\_2.2.1 Wyszukanie produktu przy użyciu jego kategorii  F\_2.2.1 Wyszukanie produktu przy użyciu jego producenta  F\_2.3 Założenie konta  F\_2.3.1 Wprowadzenie imienia  F\_2.3.2 Wprowadzenie nazwiska  F\_2.3.3 Wprowadzenie miasta  F\_2.3.4 Wprowadzenie kodu pocztowego  F\_2.3.5 Wprowadzenie adresu zamieszkania  F\_2.3.6 Wprowadzenie adresu email  F\_2.3.7 Wprowadzenie numeru telefonu |

|  |  |
| --- | --- |
| **Identyfikator** | **Opis** |
| F\_3 | **Funkcje Użytkownika:**  F\_3.1 Zalogowanie się  F\_3.2 Reset hasła  F\_3.3 Zamówienia  F\_3.3.1 Dodanie produktu do zamówienia  F\_3.3.2 Usunięcie produktu z zamówienia  F\_3.3.3 Możliwość zmiany ilości konkretnego produktu  F\_3.3.4 Wybór metody płatności  F\_3.3.4 Możliwość wprowadzenia daty realizacji jeśli jest to planowane zlecenie  F\_3.3.5 Sprawdzenie statusu zamówienia  F\_3.3.5 Wyświetlenie listy wszystkich zamówień  F\_3.3.6 Stworzenie nowego zamówienia po przez duplikację historycznego  F\_3.3.7 Płatność |

|  |  |
| --- | --- |
| **Identyfikator** | **Opis** |
| F\_4 | **Funkcje Kuriera:**  F\_4.1 Zalogowanie się  F\_4.2 Reset hasła  F\_4.3 Zamówienia  F\_4.3.1 Podjęcie zamówienia  F\_4.3.2 Porzucenie zamówienia  F\_4.3.3 Zmiana statusu zlecenia |

#### **Diagram przypadków użycia – Mateusz Rzewnicki**



#### **Wymagania pozafunkcjonalne – Mateusz Rzewnicki**

Wymagania pozafunkcjonalne opisywać mogą charakterystyki i właściwości systemu. Definiują one ograniczenia dotyczące projektowania, implementacji lub ram, jak system ma działać.

Model Standardu ISO/IEC 25010 posłuży do opracowań wymagań pozafunkcjonalnych Systemu Integracji Przedsiębiorstwa. Składa się on z ośmiu jakościowych cech, które określają właściwości oprogramowania. Aby możliwa była późniejsza ocena jakości, cechy definiowane są jako stopień, w jakim projektowany system spełnia założenia. Wszystkie cechy w modelu są podzielone na kategorie, które zawierają atrybuty:

1. Funkcjonalne dopasowanie (ang. Functional Suitability) – funkcjonalna kompletność, poprawność i odpowiedniość w funkcjonowaniu.
2. Wydajność (ang. Performance Efficiency) – charakterystyka czasowa, zużycie zasobów oraz oczekiwana wydajność.
3. Kompatybilność (ang. Compatibility) – współistnienie, czyli współdziałanie z innymi produktami w tym samym środowisku bez wpływu na ich zachowanie, interoperacyjność.
4. Użyteczność (ang. Usability) – rozpoznawalność zastosowania, łatwość uczenia się, operowania, ochrona użytkownika przed błędami, estetyka interfejsu użytkownika, dostępność.
5. Niezawodność (ang. Reliability) – dojrzałość, dostępność techniczna, odporność na wady oraz odtwarzalność w przypadku awarii.
6. Bezpieczeństwo (ang. Security) – poufność, integralność, niezaprzeczalność, identyfikowalność, autentyczność.
7. Łatwość utrzymania (and. Maintainability) – modułowość, łatwość ponownego użycia, analizy, testowania oraz modyfikowania.
8. Przenośność (ang. Portability) – łatwość instalacji, adaptacji do różnych środowisk, zamiany na inny produkt do realizacji tego samego celu.

W tabeli poniżej zostały przedstawione wymagania pozafunkcjonalne systemu według schematu:

* Identyfikator – w formacie WPXX gdzie XX jest numerem wymagania
* Opis – wyjaśnienie wymagania.
* Kategoria – jedna z kategorii standardu ISO/EIC 25010.

| Identyfikator | Opis | Kategoria |
| --- | --- | --- |
| WP01 | System powinien działać w najpopularniejszych przeglądarkach: Chrome 29.0+, Firefox 28.0+, Safari 9.0+, Edge 11.0+, Opera 17.0+. | przenośność |
| WP02 | System powinien dostosowywać swój interfejs w zależności od urządzenia, na którym jest uruchamiany. | użyteczność |
| WP03 | Hasło użytkownika powinno składać się z co najmniej 8 liter, co najmniej jednego znaku specjalnego oraz co najmniej jednej wielkiej litery. | bezpieczeństwo |
| WP04 | System powinien generować możliwie najmniejsze koszty licencyjne. | łatwość utrzymania |
| WP05 | System powinien mieć czytelny i intuicyjny interfejs. | użyteczność |
| WP06 | System powinien wyraźnie zaznaczać pola obowiązkowe w formularzach. | użyteczność |
| WP07 | System nie powinien blokować działania zewnętrznych źródeł danych poprzez wielokrotne wywoływanie zapytań. | wydajność |
| WP08 | Baza danych musi być łatwo rozszerzalna o nowe produkty. | funkcjonalne dopasowanie |
| WP09 | System powinien wyraźnie sygnalizować krytyczne błędy użytkownikowi. | użyteczność |
| WP10 | System musi być w stanie obsługiwać co najmniej 30 użytkowników w tym samym czasie z wysoką wydajnością | wydajność |
| WP11 | Rozwiązanie powinno być stworzone w architekturze wielowarstwowej | funkcjonalne dopasowanie |
| WP12 | Możliwość integracji z systemami zewnętrznymi | kompatybilność |

### Modele danych – Michał Baca



Rysunek 1 Diagram ERD.

Tabela poniżej przedstawia opis encji.

|  |  |  |
| --- | --- | --- |
| Nr | Nazwa encji | Opis |
| 1 | Rola | Encja przechowująca uprawnienia jakie może posiadać użytkownik systemu. |
| 2 | Użytkownik | Użytkownicy systemu wraz z podstawowymi danymi dotyczącymi użytkownika. |
| 3 | Koszyk | Koszyk klienta. |
| 4 | Kurier | Dane dotyczące kuriera. |
| 5 | Zamówienia | Encja przechowująca dane zamówienia. |
| 6 | Status | Statusy jakie może uzyskać zamówienie. |
| 7 | Produkt | Produkty dostępne w systemie. |
| 8 | Producent | Zbiór producentów produktów. |
| 9 | Kategoria | Kategorie produktów. |
| 10 | Jednostka miary | Słownik jednostek miary. |

Tabela 1 Opis encji programu Pandronka.

Zbiór relacji został przedstawiony w poniższej tabeli.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Encja I | Encja II | Rodzaj związku | Opis |
| 1 | Rola | Użytkownik | 1:N | Jedna rola może być przyporządkowana do wielu użytkowników. |
| 2 | Użytkownik | Kurier | 1:N | Użytkownik może być kurierem. |
| 3 | Użytkownik | Zamówienia | 1:N | Jeden użytkownik może złożyć wiele zamówień. |
| 4 | Użytkownik | Koszyk | 1:N | Jeden użytkownik może posiadać kilka koszyków. |
| 5 | Kurier | Zamówienia | 1:N | Jeden kurier realizuje wiele zamówień. |
| 6 | Zamówienia | Status | N:1 | Jeden status może mieć wiele zamówień. Zamówienie może mieć tylko jeden status. |
| 7 | Zamówienia | Koszyk | N:1 | Koszyk może przynależeć do zamówienia. |
| 8 | Koszyk | Produkt | N:N | Wiele koszyków może posiadać kilka produktów. |
| 9 | Producent | Produkt | 1:N | Jeden producent może przynależeć do wielu produktów. |
| 10 | Kategoria | Produkt | 1:N | Jedna kategoria może przynależeć do wielu produktów. |
| 11 | JednostkaMiary | Produkt | 1:N | Jednostka miary może przynależeć do kilku produktów. |

### Model architektury systemu – Rafał Sawicki

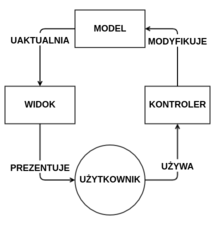
System dla obsługi zamówień zakupów online Pandronka będzie aplikacja webową. Klient aplikacji będzie uruchomiony na serwerze w usłudze IIS, dzięki czemu nie jest wymagana żadna usługa instalacyjna na stanowiskach użytkowników.

Do jego uruchomienia wystarczy przeglądarka stron internetowych.

Baza danych będzie funkcjonować na serwerze SQL.

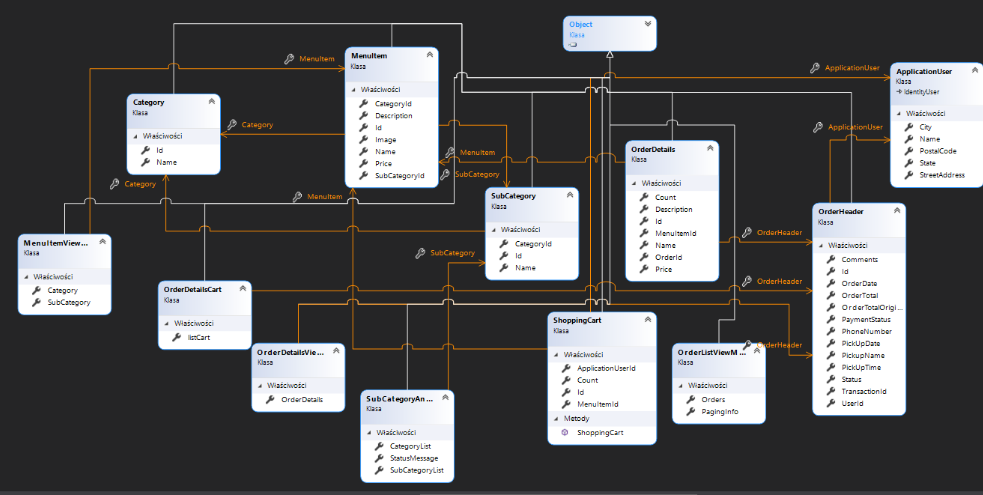
Rys. 3. Model architektury systemu.

Wzór architektoniczny zastosowany to MVC (Model-view-controller) jest tzw. wzorcem projektowym, stosowanym przy tworzeniu nowoczesnych systemów informatycznych. **Model** – jest pewną reprezentacją problemu bądź logiki aplikacji.

* 
* Zmiana danych przez użytkownika odbywa się za pomocą kontrolera, ten modyfikuje model, model odświeża widok, a informacja z widoku dociera do użytkownika.
* **Widok** – opisuje, jak wyświetlić pewną część modelu w ramach interfejsu użytkownika. Może składać się z podwidoków odpowiedzialnych za mniejsze części interfejsu.
* **Kontroler** – przyjmuje dane wejściowe od użytkownika i reaguje na jego poczynania, zarządzając aktualizacje modelu oraz odświeżenie widoków.

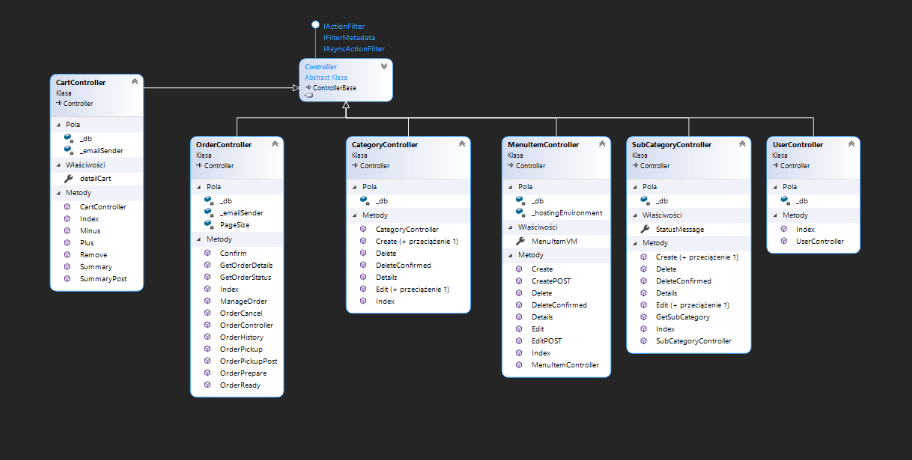
Projekt architektury systemu

## Diagramy klas – Michał Baca

Diagram klas ze względu na lepszą czytelność został podzielony na dwie warstwy. Warstwa kontrolerów będzie komunikować się z modelem systemu za pomocą wstrzykiwania zależności (ang. dependency injection). W konstruktorze kontrolera wywoływane jest przypisanie klasy ApplicationDbContext do stworzonego w danej klasie pola. Następnie dzięki stworzonemu kontekstowi modelu danych, można wykonywać operacje na danych w kontrolerze. Poniższy diagram przedstawia diagram klas modelu oraz relacje pomiędzy nimi. 

Rysunek 2 Diagram klas modelu systemu Pandronka.

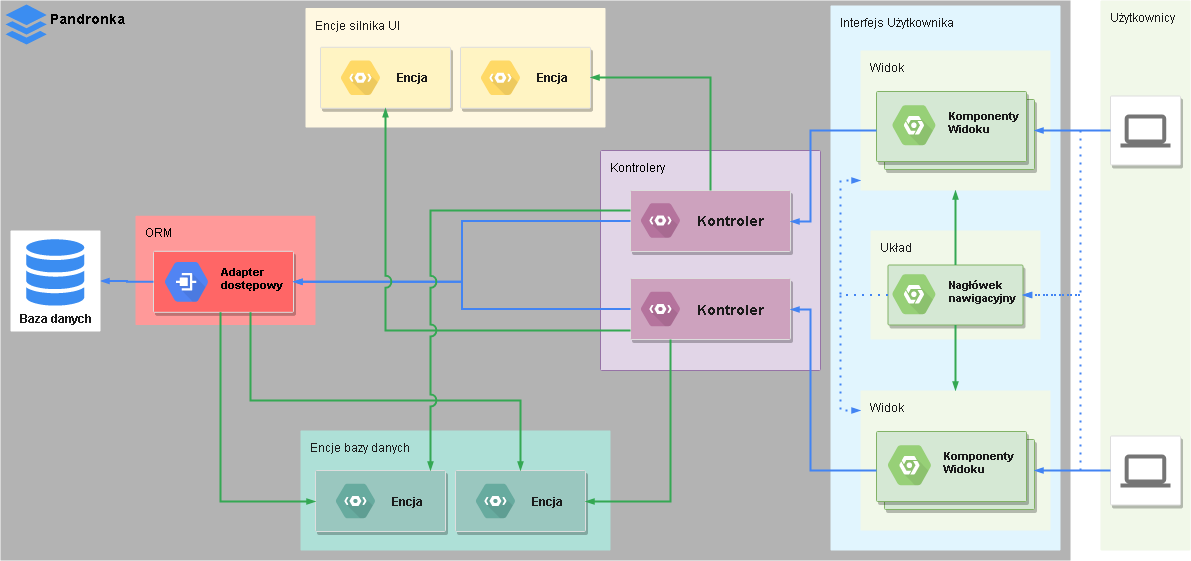
Diagram kontrolerów projektowanego systemu zostały przedstawione poniżej.



W tabeli zostały opisane klasy projektowanego systemu.

| Nazwa klasy | Opis |
| --- | --- |
| MenuItemController | Klasa kontrolera umożliwiająca zarządzanie produktami. Wymagania funkcjonalne realizowane za pomocą kontrolera : F\_1.3. |
| HomeController | Klasa kontrolera umożliwiająca wyświetlania ilości produktów w koszyku oraz głównego menu. |
| SubCategoryController | Klasa kontrolera umożliwiająca dodanie podkategorii. |
| UserController | Klasa kontrolera umożliwiająca odblokowanie użytkownika. |
| CategoryController | Klasa kontrolera umożliwiająca zarządzanie kategoriami produktów. Wymaganie funkcjonalne F\_1.3.6 |
| CartController | Klasa kontrolera umożliwiająca realizację koszyka. Wymaganie funkcjonalne F\_2.1. |
| OrderController | Klasa kontrolera odpowiadająca za obsługę zamówień. Wymagania funkcjonalne F\_3.3. |
| Program | Klasa frameworska ASP.NET Core służaca do inicjalizacji hosta aplikacji. |
| Startup | Klasa framewroka ASP.NET Core służącą do rejestracji usług oraz konfiguracji m.in. routingu. |
| OrderDetails | Klasa OrderDetails jest reprezentacją definicji tablicy szczegóły zamówienia, jej obiekt natomiast pojedynczego rekordu tabeli. Jest to tablica słownika umożliwiająca zdefiniowanie śledzonego później procesu za pomocą identyfikatorów systemowych. |
| OrderHeaders | Klasa OrderHeaders jest reprezentacją definicji tablicy Zamówienia, jej obiekt natomiast pojedynczego rekordu tabeli. Jest ona tablicą konsolidującą wiedzy z kilku tabel, umożliwia przechowywanie danych z realizacji założeń biznesowych. Po za kolumnami identyfikatora i kwoty zawiera ona 3 powiązania: Użytkownik, Status i Koszyk. |
| ApplicationUser | Klasa ApplicationUser jest reprezentacją definicji tablicy ApplicationUser, jej obiekt natomiast pojedynczego rekordu tabeli. Przechowuje ona dane na temat użytkownika systemu i umożliwia po przez przechowywane dane teleadresowe realizację założeń biznesowych po przez wskazanie celu dostarczenia zamówienia. |
| ShoppingCart | Klasa ShoppingCart jest reprezentacją definicji tablicy Koszyk, jej obiekt natomiast pojedynczego rekordu tabeli. Przechowuje ona dane o przypisaniu koszyka do użytkownika po przez referencje. |
| MenuItem | Klasa MenuItem jest reprezentacją definicji tablicy MenuItem, jej obiekt natomiast pojedynczego rekordu tabeli. Przechowuje ona dane produktu w systemie takie jak jego nazwa, dostępność lub cenę oraz po przez 3 referencje jednostki miary, kategorii i producenta. Umożliwia to realizację założeń biznesowych po przez przechowywanie produktów oferowanych przez firmę oraz funkcjonalności sortowania po przez użycie identyfikatorów systemowych danych słownikowych. |
| Category | Klasa Category jest reprezentacją definicji tablicy Kategoria, jej obiekt natomiast pojedynczego rekordu tabeli. Jest to tablica słownikowa umożliwiająca zdefiniowanie produktu dzięki czemu możliwe jest sprawne ich wyszukiwanie przez użytkowników. |
| ApplicationDbContext | Jest klasą dostępowa do bazy danych, która przechowuje referencje do wszystkich wyżej wymienionych modeli, co umożliwia odwołanie się do nich po przez model obiektowy zgodnie z paradygmatem bazy obiektowo-relacyjnej i ich mapowaniem. |

## Schemat warstw – Mateusz Rzewnicki

Poniższy schemat przedstawia warstwy na wysokim poziomie uogólnienia, szczegółowe schematy każdego modułu zostały przedstawione w następnym podrozdziale (Schematy modułów). Szczegółowe prezentacje nie zawierają jedynie każdorazowo importowanego układu ze względu na czytelność

Wyjaśnienie pojęć:

* Encje silnika UI – są to modele widoku do których przekazywane będą dane bazy danych.
* Encje bazy danych – są to modele bazy danych których klasy są reprezentacją definicji tabeli a obiekty wierszy.

## Schematy modułów – Mateusz Rzewnicki

Ze względu na czytelność każdy moduł przedstawiany jest osobno.

### Moduł Produktu

### Moduł Kuriera

### Moduły Producentów, Kategorii i Jednostki

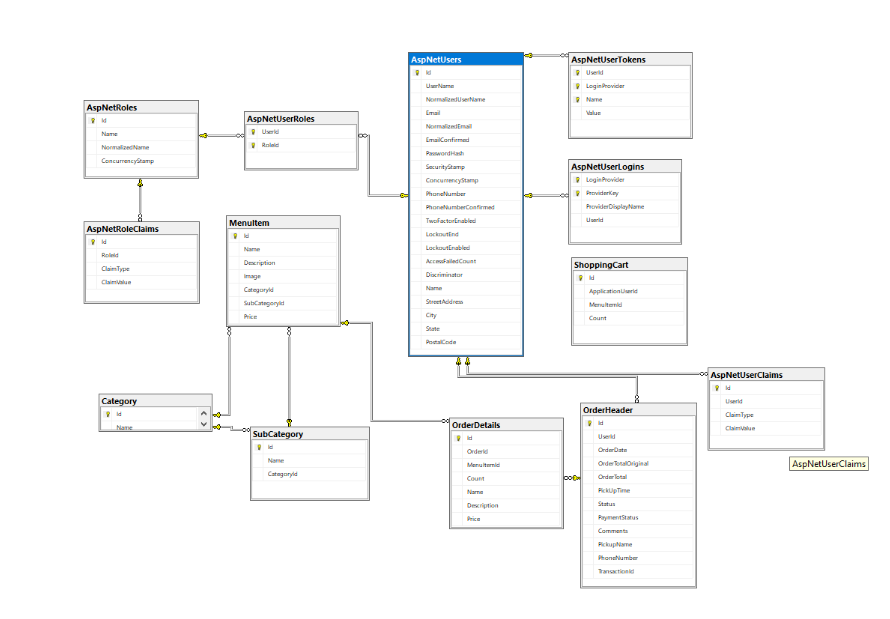
### Moduł Zamówienia

### Moduł Koszyka

### Moduł Administratora

### Moduł Kont

Projekt bazy danych – Michał Baca



W poniższej tabeli zostały opisane klasy modelu systemu.

|  |  |  |
| --- | --- | --- |
| Nr | Nazwa tabeli | Opis |
| 1 | ShoppingCart | Koszyk klienta. |
| 2 | SubCategory | Podkategoria produktu |
| 3 | OrderHeader | Encja przechowująca dane zamówienia. |
| 4 | OrderDetails | Szczegóły zamówienia |
| 5 | MenuItem | Produkty dostępne w systemie. |
| 6 | AspNetUserTokens | Tabela przechowująca tokeny użytkowników. Nie znajduje się w modelu danych. Powstałą w skutek użycia mechanizmu Identity. |
| 7 | Category | Kategorie produktów. |
| 8 | AspNetUsers | Tabela przechowująca użytkowników. W modelu danych odpowiada nazwie Użytkownik. |
| 9 | AspNetRoleClaims | Encja przechowująca oświadczenia dla ról. Nie znajduje się w modelu danych. Powstała w skutek użycia mechanizmu Identity. |
| 10 | AspNetRoles | Odpowiada encji Role w modelu danych. Zawiera role użytkowników systemu. |
| 11 | AspNetUserClaims | Tabela zawiera oświadczenia użytkowników. Nie znajduje się w modelu danych. Powstała w skutek użycia mechanizmu Identity. |
| 12 | AspNetUserLogins | Tabela przechowująca loginy użytkowników. Nie znajduje się w modelu danych. Powstałą w skutek użycia mechanizmu Identity. |
| 13 | AspNetUserRoles | Tabela zawierająca uprawnienia użytkowników. Nie znajduje się w modelu danych. Powstałą w skutek użycia mechanizmu Identity. |

Zbiór relacji został przedstawiony w poniższej tabeli.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Encja I | Encja II | Rodzaj związku | Opis |
| 1 | OrderHeaders | ShoppingCart | N:1 | Koszyk może przynależeć do zamówienia. |
| 2 | ShoppingCart | MeniUtem | N:N | Wiele koszyków może posiadać kilka produktów. Tabela pomocnicza Kosz\_Prod. |
| 3 | Category | MenuITem | 1:N | Jedna kategoria może przynależeć do wielu produktów. |
| 4 | AspNetUsers | OrderHeaders | 1:N | Użytkownik może mieć wiele zamówień. |
| 5 | AspNetUsers | ShoppingCart | 1:1 | Użytkownik może mieć jeden koszyk. |
| 6 | AspNetUsers | AspNetUserClaims | 1:N | Użytkownik może mieć wiele oświadczeń. |
| 70 | AspNetUsers | AspNetUserLogins | 1:N | Użytkownik może mieć wiele loginów. |
| 8 | AspNetUsers | AspNetUserTokens | 1:N | Użytkownik możę mieć wiele tokenów. |
| 9 | AspNetUsers | AspNetRoles | N:N | Użytkownik może mieć wiele ról. |
| 10 | AspNetRoles | AspNetRoleClaims | 1:N | Rola może mieć wiele oświadczeń. |

Poniżej przedstawione zostaną tabele wraz z opisem i typ danych.

Tabela AspNetUserTokens.

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | UserId | Text | Identyfikator użytkownika |
| 2 | LoginProvider | Text | Dostawca logowania |
| 3 | Name | Text | Nazwa tokenów |
| 4 | Value | Text | Wartość tokenu |

Tabela AspNetUserRoles

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | UserId | Text | Identyfikator użytkownika |
| 2 | RoleId | Text | Identyfikator roli |

Tabela AspNetUserLogins

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | LoginProvider(PK) | Text | Dostawca logowania |
| 2 | ProviderKey | Text | Klucz dostawcy logowania |
| 3 | ProviderDisplayName | Text | Nazwa dostawcy logowania |
| 4 | UserId | Text | Identyfikator użytkownika |

Tabela AspNetUserClaims

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id(PK) | Int | Identyfikator oświadczeń użytkownika |
| 2 | UserId | Text | Identyfikator użytkownika |
| 3 | ClaimType | Text | Typ oświadczenia |
| 4 | ClaimValue | Text | Wartość oświadczenia |

Tabela AspNetRoleClaims

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id(PK) | Int | Identyfikator oświadczeń roli |
| 2 | RoleId | Text | Identyfikator roli |
| 3 | ClaimType | Text | Typ oświadczenia |
| 4 | ClaimValue | Text | Wartość oświadczenia |

Tabela AspNetUserClaims

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id(PK) | Int | Identyfikator oświadczenia |
| 2 | UserId | Text | Identyfikator użytkownika |
| 3 | ClaimType | Text | Typ oświadczenia |
| 4 | ClaimValue | Text | Wartość oświadczenia. |

Tabela AspNetRoles

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id(PK) | Int | Klucz podstawowy tabeli. Identyfikator roli w systemie. |
| 2 | Name | Varchar(256) | Nazwa roli użytkownika |
| 3 | NormalizedName | Varchar(256) | Nazwa roli zapisana kapitalkami |
| 4 | ConcurrencyStamp | Text | Pieczęć bezpieczeństwa zapobiegają nadpisaniu danych przez kilku użytkowników. |

Tabela AspNetUsers

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id(PK) | Text | Klucz podstawowy tabeli. Numer identyfikacyjny użytkownika. |
| 2 | UserName | Varchar(256) | Nazwa użytkownika |
| 3 | NormalizedUserName | Varchar(256) | Nazwa użytkownika zapisana kapitalikami |
| 4 | Email | Varchar(256) | Adres email. |
| 5 | NormalizedEmail | Varchar(256) | Adres email zapisany kapitalikami. |
| 6 | EmailConfirmed | Bit | Potwierdzenie adresu mailowego. |
| 7 | PasswordHash | Text | Zaszyfrowane hasło. |
| 8 | SecurityStamp | Text | Pieczęć bezpieczeństwa |
| 9 | ConcurrencyStamp | Text | Pieczęć zabezpieczająca równoczesne zapisane danych w tym samym czasie przez wielu użytkowników. |
| 10 | PhoneNumber | Text | Numer telefonu |
| 11 | PhoneNumberConfirmed | Text | Potwierdzenie numeru telefonu |
| 12 | TwoFactorEnabled | bit | Potwierdzenie aktywacji autoryzacji dwustopniowej |
| 13 | LockoutEnd | Datetime | Data zakończenia blokady konta |
| 14 | LockoutEnabled | Bit | Potwierdzenie blokady konta. |
| 15 | AccessFailedCount | Int | Licznik nieudanych logowań. |
| 16 | Name | Nvarchar(256) | Imię |
| 17 | StreetAdresss | Nvarchar(256) | Ulica |
| 18 | City | Nvarchar(256) | Miejscowość |
| 19 | State | Nvarchar(256) | Poczta |
| 20 | PostalCode | Nvarchar(256) | Kod pocztowy |

Tabela Category

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id | Int | Identyfikator kategorii |
| 2 | Name | Text | Nazwa kategorii. |

Tabela ShoppingCart

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id | Int | Identyfikator koszyka |
| 2 | UzytkownikId | text | Identyfikator użykownika |

Tabela produkt

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id | Int | Identyfikator produktu |
| 2 | Name | Text | Nazwa produktu |
| 3 | Description | Text | Opis produktu |
| 4 | Image | Text | Url obrazu |
| 5 | CategoryId | int | Identyfikator kategorii |
| 6 | SubCategoryId | Int | Identyfikator podkategorii |
| 7 | Cena | double | Cena produktu |

Tabela OrderHeader

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id | Int | Identyfikator zamówienia |
| 2 | UserId | String | Identyfikator użytkownika |
| 3 | OrderDate | Datetime | Data zamówienia |
| 4 | OrderTotal | Double | Całość zamówienia |
| 5 | PickupDate | Datetime | Data odbioru |
| 6 | Status | String | Status zamówienia |
| 7 | PaymentStatus | String | Status płatności |
| 8 | Comments | String | Komentarz |
| 9 | PickupName | String | Dostawca |
| 10 | PhoneNUmber | String | Numer telefonu |
| 11 | TransactionId | String | Identyfikator transakcji |

Tabela OrderDetails

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id | Int | Identyfikator szczegółów zamówienia |
| 2 | OrderId | Int | Identyfikator zamówienia |
| 3 | MenuItemId | Int | Identyfikator produktu |
| 4 | Count | Int | Ilość |
| 5 | Name | String | Nazwa |
| 6 | Description | String | Opis |
| 7 | Price | Doubel | Cena |

Tabela OrderDetails

|  |  |  |  |
| --- | --- | --- | --- |
| Lp | Nazwa | Typ | Opis |
| 1 | Id | Int | Identyfikator |
| 2 | Name | String | Nazwa podkategorii |
| 3 | CategoryId | Int | Kategoria |

Projekt algorytmów – Mateusz Rzewnicki

Modele struktury opisują – na różnych poziomach abstrakcji – elementy składowe oprogramowania oraz ich statyczne związki, nie wyjaśniają natomiast sposobu, w jaki te elementy współpracują ze sobą podczas wykonywania programu. Docelowym wzorcem zachowania programów jest model przypadków użycia. Zadaniem modeli współdziałania jest pokazanie, jak zrealizować wszystkie scenariusze przypadków użycia za pomocą obiektów i komponentów zdefiniowanych w modelach struktury. Narzędziami modelowania współdziałania są diagramy sekwencji i komunikacji.

Zarówno treść, jak i forma diagramu sekwencji, zwanego też diagramem przebiegu, nawiązują do wykresu obrazującego przebieg zdarzań zachodzących w czasie działania systemu. Czas na diagramie biegnie z góry na dół, rysując pionowe linie życia obiektów rozmieszczonych wzdłuż górnego brzegu rysunku. Między tymi liniami są prowadzone strzałki obrazujące komunikaty przekazywane między obiektami e celu wywołania metod. Cały wykres pokazuje jeden scenariusz wykonania pewnej operacji, np. przypadków użycia, przez obiekty (nie klasy) istniejące i współpracujące w tym celu podczas wykonania programu.

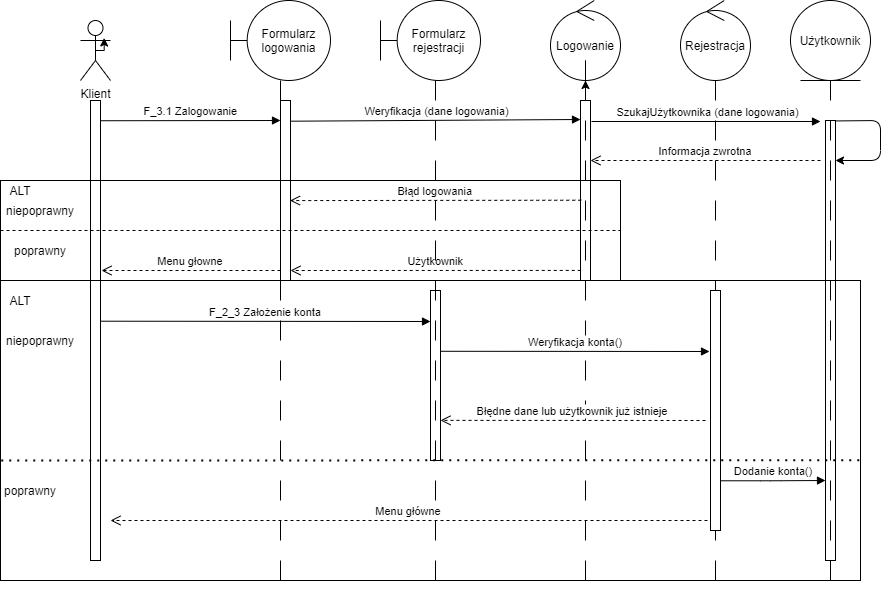
Podczas swojego życia obiekt ma okresy aktywności, gdy wykonuje się jedna z jego metod, i bezczynności, gdy żadna z jego metod nie jest wykonywana. Okresy aktywności obiektu można pokazać na diagramie za pomocą pasów rysowanych na właściwych odcinkach jego linii życia. Wywołanie własnych metod obiektu zaznacza się za pomocą wywołań skierowanych do własnej linii życia.

Komunikaty przekazywane między obiektami mogą mieć semantykę synchroniczną, zgodnie z którą obiekt wywołujący zawiesza działanie do czasu otrzymania odpowiedzi, lub asynchroniczną, zgodnie z którą obiekt wywołujący i wywoływany działają równolegle. Przypadek zaznaczony graficznie strzałką z pełnym grotem, opisuje wywołanie metody innego obiektu w ramach sekwencyjnego programu. Drugi przypadek o reprezentacji strzałki otwartej, opisuje współdziałanie obiektów lub komponentów w systemie współbieżnym.

Etykieta komunikatu określa sygnaturę – jeśli na rysunku jest miejsce na tak długi napis – lub przynajmniej nazwę wywoływanej metody. Etykiety strzałek zaznaczających powrót wywołanej metody mogą opisywać zwracane wartości. Komunikaty które tworzą lub usuwają obiekty, można opatrzeć stereotypem *<<create>>*, *<<new>>* lub *<<destroy>>*.

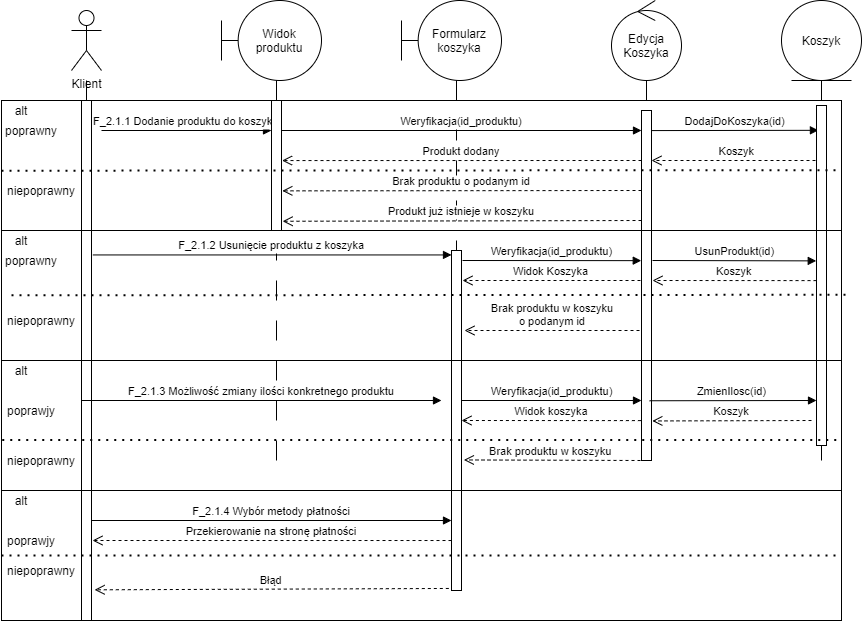
## Diagram sekwencji logowania i rejestracji – Michał Baca

Realizacja funkcjonalności F\_2.3

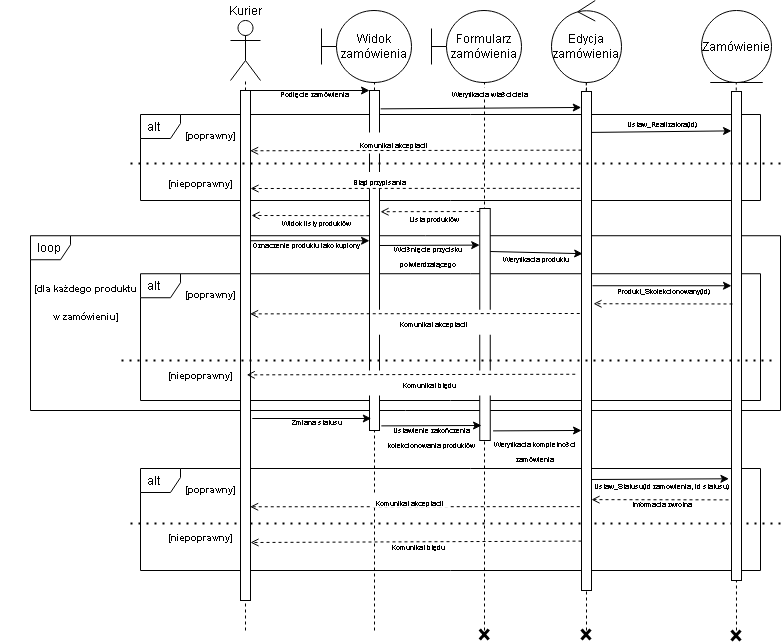


## Diagram sekwencji obsługi koszyka – Michał Baca

Realizacja funkcjonalności 2.1

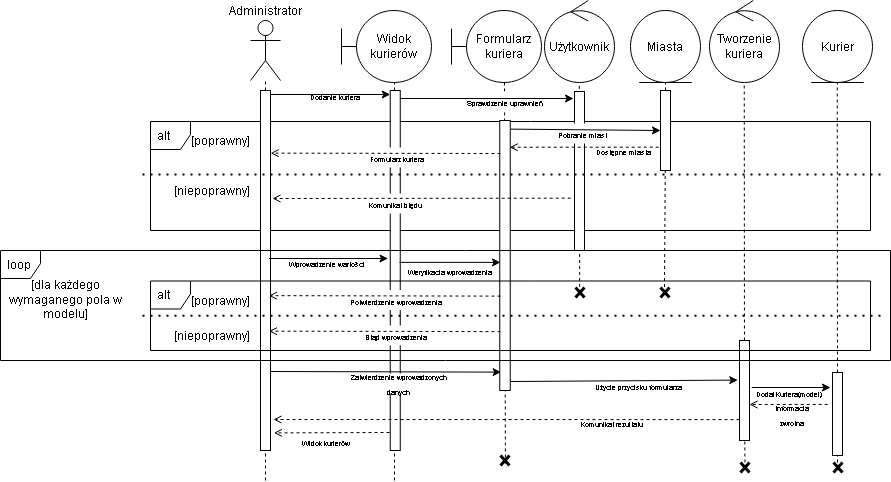


## Diagram sekwencji realizacji zamówienia przez kuriera – Mateusz Rzewnicki

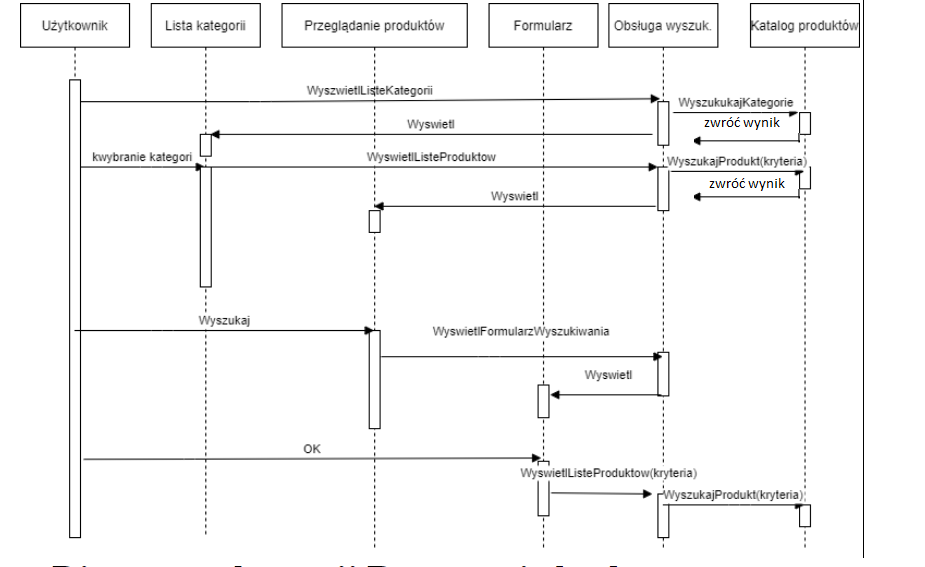
Realizacja funkcjonalności F\_4.3

## Diagram sekwencji wprowadzenia kuriera – Mateusz Rzewnicki

Realizacja funkcjonalności F1.2

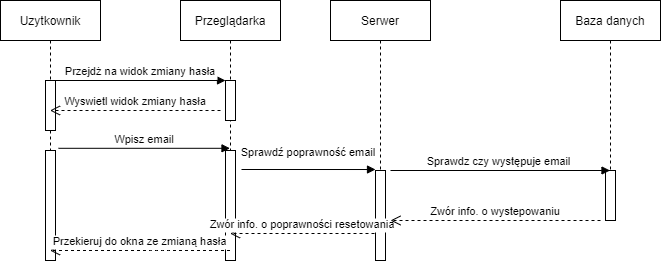


**Diagram sekwencji wyszukiwanie** – Rafał Sawicki

Realizacja funkcjonalności F2.2

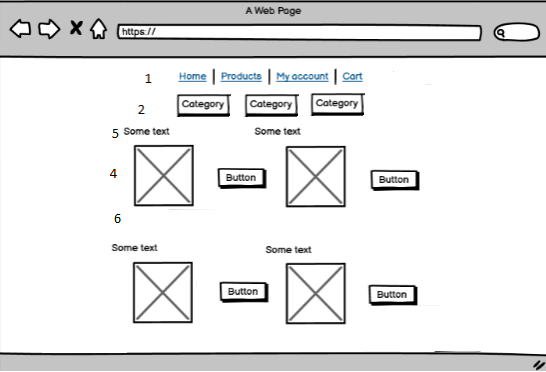
**Diagram sekwencji Resetowaie hasła** – Rafał Sawicki

Realizacja funkcjonalności F3.2



Projekt interfejsu użytkownika - Rafał Sawicki

## Homepage



1 Menu główne

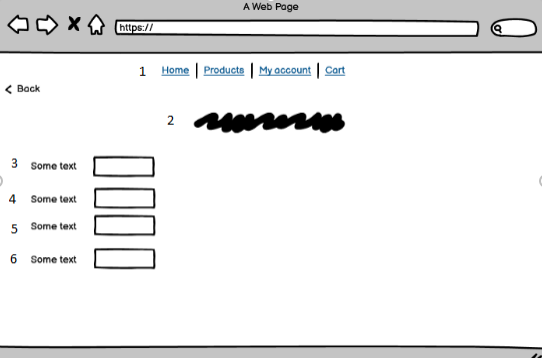
2 Textboxy – kategorie

3 Button – Szczegoly produktu

4 Img– zdjecie

5 Text – opis

## Panel użytkownika



1 Menu główne

2 Title - tytuł

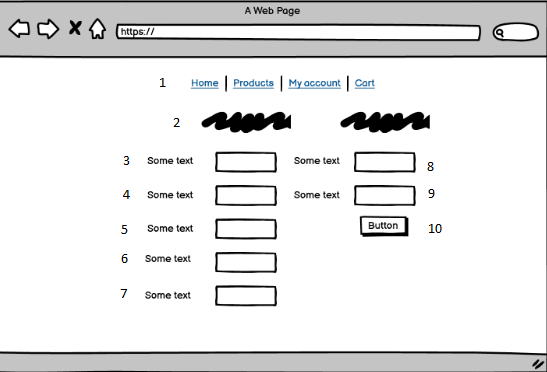
3 Label, Textbox – dane użytkownika

4 Label, Textbox – dane użytkownika

5 Label, Textbox – dane użytkownika

6 Label, Textbox – dane użytkownika

## Nowy produkt



1.Meny główne

2.Title - Tytuł

3.Label, Textbox – dane nowego produktu

4. Label, Textbox – dane nowego produktu

5. Label, Textbox – dane nowego produktu

6. Label, Textbox – dane nowego produktu

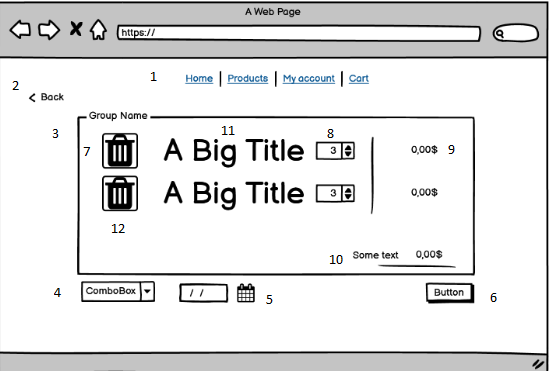
7. Label, Textbox – dane nowego produktu

8. Label, Textbox – dane nowego produktu

9. Label, Textbox – dane nowego produktu

10.Button - zapisz danego nowego produktu

## Koszyk



Menu główne

Anchor - cofnięcie na poprzednią stronę

Panel – panel zamówienia klienta

Combobox - wybór sposobu dostawy

DatePicker – ustawienie terminu dostawy

Button – zapis zamówienia

Row – Produkt opis

NumbChanger – zmiana ilości

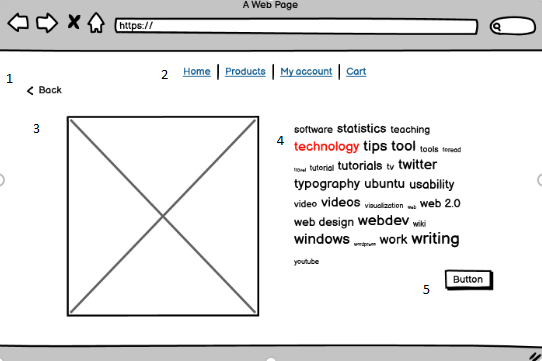
Text – koszt produktu

Label – kwota do zapłaty

Title – nazwa produktu

Button – usun z koszyka

## Informacje o produkcie



1.Anchor - cofnięcie do poprzedniej strony

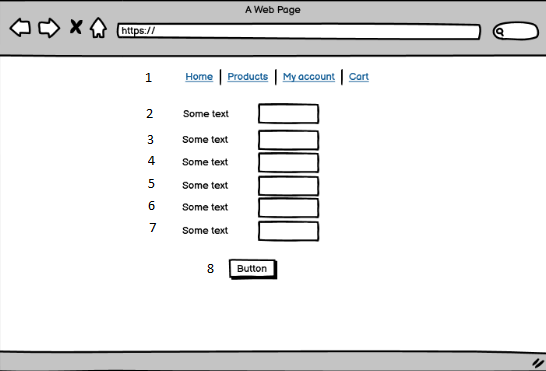
2. Menu głowne

3.Image - zdjęcie produktu

4.Text - opis produktu

5.Button - dodaj do koszyka

## Rejestracja



Menu główne

Label, Textbox – dane nowego użytkownika

Label, Textbox – dane nowego użytkownika

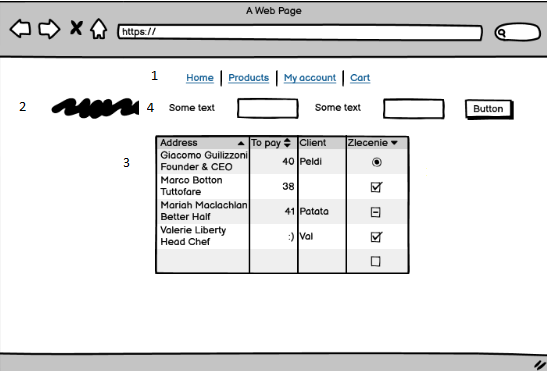
Label, Textbox – dane nowego użytkownika

Label, Textbox – dane nowego użytkownika

Label, Textbox – dane nowego użytkownika

Label, Textbox – dane nowego użytkownika

Button - Rejestracja

Kurier  


1.Menu głowne

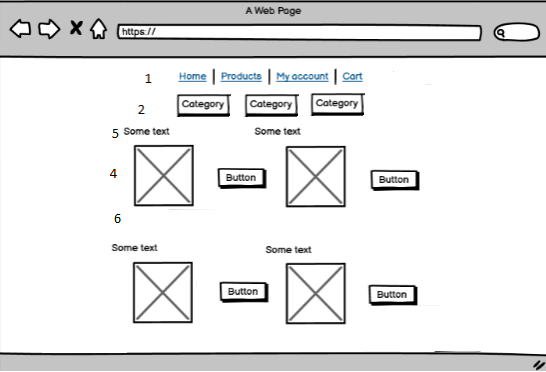
2. Title – tytul strony

3. Grid z zamówieniami

4. Wyszukiwarka

5. Button – wyszukaj przycisk

## Produkty



1Menu główne

2 Combobox - Wybór kategorii

3 Button – dodaj nowy produkt

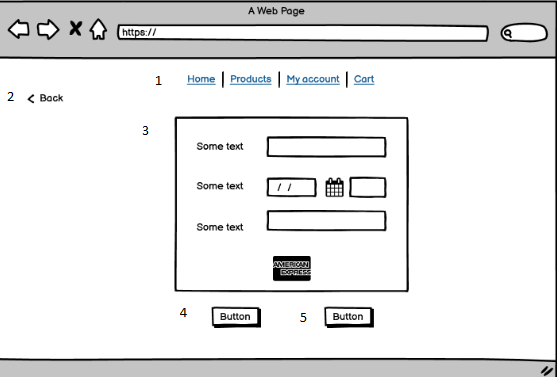
4 Img - zdjęcie produtu

5 Title – nazwa produktu

6 Button – dodaj do koszyka

7 Textbox – wyszukiwarka

## Płatność kartą



1 Menu główne

2 Anchor - cofnięcie do poprzedniego widoku

3 Panel – wprowadzenie danych karty płatniczej

4 Button – anulowanie zamówienia i zapłaty

5 Button – potwierdzenie płatności

Projekt testów kontrolnych funkcjonalne - Rafał Sawicki

|  |  |  |
| --- | --- | --- |
| **Przypadek testowy wysokiego poziomu** | **Przypadek testowy niskiego poziomu** | **Statut testu** |
| **TYTUŁ: Sprawdzenie wyszukiwania ciągu „name”**  **WARUNEK POCZĄTKOWY: Strona Odbior zamowienia uruchomiona dla kuriera**  **OCZEKIWANY REZULTAT: Na liście wyników wyszukiwania pojawiają się wyświetlone elementy strony z ciągiem „ name”**  **WARUNEK KOŃCOWY: Wyświetlona strona z wynikami wyszukiwania** | **KROKI: 1. Zaznacz obszar wyszukiwarki, 2. Wprowadź ciąg „name”,** **3. Naciśnij przycisk „Szukaj”** | **TYTUŁ: Sprawdzenie wyszukiwania**  **Czas: sekunda** |
| **TYTUŁ: Dostęp do funkcji systemu jest ograniczony.**  **POCZĄTKOWY: Uruchomiona strona**  **OCZEKIWANY REZULTAT: Na stronie wyświetlone jedynie informację dla danego użytkownika.**  **WARUNEK KOŃCOWY: Wyświetlone informacje pokrywają się z funkcją użytkownika.** | **KROKI: 1. Zaloguj się na konto Kuriera,**  **2. W menu widoczne tylko pole “Zamowienie”** | **TYTUŁ: Sprawdzenie ograniczenia dostępności** |
| **TYTUŁ: Wypełnione pola w procesie rejestracji”**  **WARUNEK POCZĄTKOWY: Strona rejestracja uruchomiona**  **OCZEKIWANY REZULTAT: Uzytkownik może założyc konto po uprzednim wypełnieniu wszystkich pól w oknie rejestracji**  **WARUNEK KOŃCOWY: Informacja o poprawnej rejestracji** | **KROKI: 1. Przejdź do widoku rejestracji nowego użytkownika, 2. Wypełnij wszystkie pola,** **3. Naciśnij przycisk „Rejestruj”** | **TYTUŁ: Walidacja pół w procesie rejestracji** |
| **TYTUŁ: Unikalność wprowadzanych danych”**  **WARUNEK POCZĄTKOWY: Strona rejestracja uruchomiona**  **OCZEKIWANY REZULTAT: Uzytkownik może założyc konto na unikalne dane email**  **WARUNEK KOŃCOWY: Wyświetlona Informacja o istniejącym już użytkowniku przypisanym do tego adresu email** | **KROKI: 1. Przejdź do widoku rejestracji nowego użytkownika, 2. Uzupełnij pole email o wartości występującej już w bazie danych** **3. Naciśnij przycisk „Rejestruj”** | **TYTUŁ: Unikalność rejestrowanych kont przez użytkowników** |
| **TYTUŁ: Weryfikacja wyświetlanych produktów na liście”**  **WARUNEK POCZĄTKOWY: Strona listy produktów wyswuetlona,**  **OCZEKIWANY REZULTAT: Każdy nowo dodany produkt dodawany jest w regularnych odstępach do poprzedniego**  **WARUNEK KOŃCOWY: Poprawne wyświetlanie nowo dodanych produktów** | **KROKI: 1. Przejdź do widoku listy produktów nowego użytkownika, 2. Naciśnij przycisk „Dodaj produkt”**  **3. Wypełnij wymagane pola dla nowego produktu,**  **4. Zweryfikuj poprawnośc wyświetlania nowo dodanego produktu** | **TYTUŁ: Weryfikacja poprawności wyświetlania nowego produktu na liście produktów** |

**Projekt testu integracyjnego - Rafał Sawicki**

wykonywany w celu wykrycia defektów w interfejsach i interakcjach pomiędzy modułami.

Utworzenie projektu dla testów integracyjnych:

EndToEnd;

Integration;

IntegrationTests.

Do projektu instalujemy paczkę Microsoft.AspNetCore.TestHost z NuGet. Po dołączeniu zależności możemy tworzyć i konfigurować serwer testowy na potrzeby testów.

Napisanie przykładowych testów w xUnit sprawdzających poprawność wywołania API dla kontrolera ProductController.

Dla żadania POST Należy stworzyć obiekt zawierający model, który dziedziczy po HttpContent.

Ucuchomić stworzone projekty testów i zweryfikowac poprawność przechodzenia.

**Projekt testów jednostkowych - Rafał Sawicki**

Wykorzystany zostanie **wzorzec AAA – Arrange, Act, Assert**.

Wzorzec ten zakłada podział testu na trzy części:

* + **Arrange – aranżacja.** W tej części testu tworzymy środowisko testowe. Tworzymy mocki oraz instancje testowanych obiektów, i definiujemy inne niezbędne element składowe.
  + **Act – odegranie.** W tym momencie następuje faktyczne użycie testowanego kodu.
  + **Assert – sprawdzenie.** Ostatnim etapem jest sprawdzenie rezultatu. Może to być sprawdzenie wyniku zwracanego przez metodę, czy upewnienie się o wywołaniu metody.

Uruchamiamy nasze testy i w test explorze powinien się pojawić zielony kolor, oznaczający, że nasze testy przeszły pozytywnie, nasz kod produkcyjny działa prawidłowo - wdrażamy na produkcję :)

**Projekt testów obciążeniowych - Rafał Sawicki**

Wykorzystaywane narzędzie - JMeter

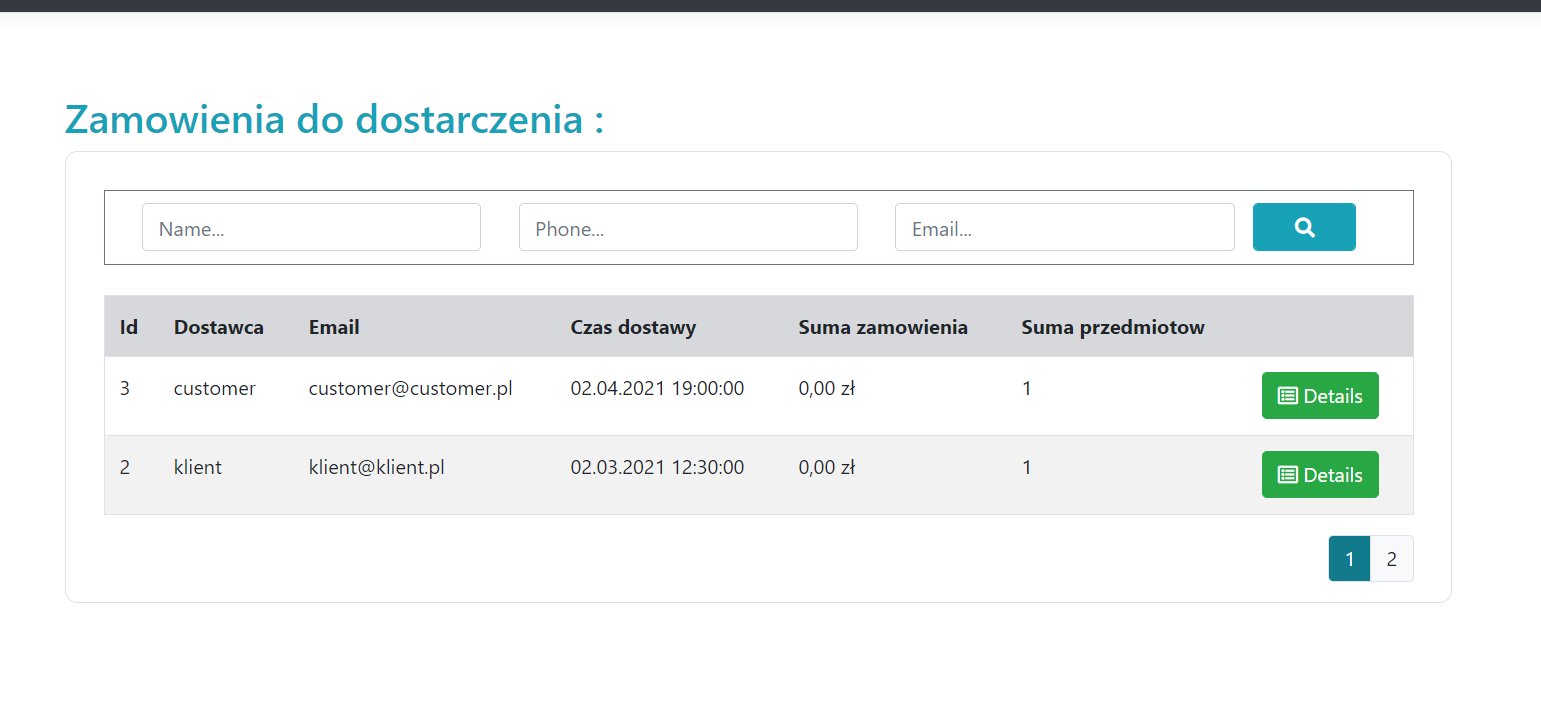
Zostanie wykonane sprawdzenia jak obciążenie aplikacji przekłada się na czas odpowiedzi. Narzędzie zostanie wykorzystane w celu zbadania, ile zapytań może obsłużyć aplikacja w określonej jednostce czasu.

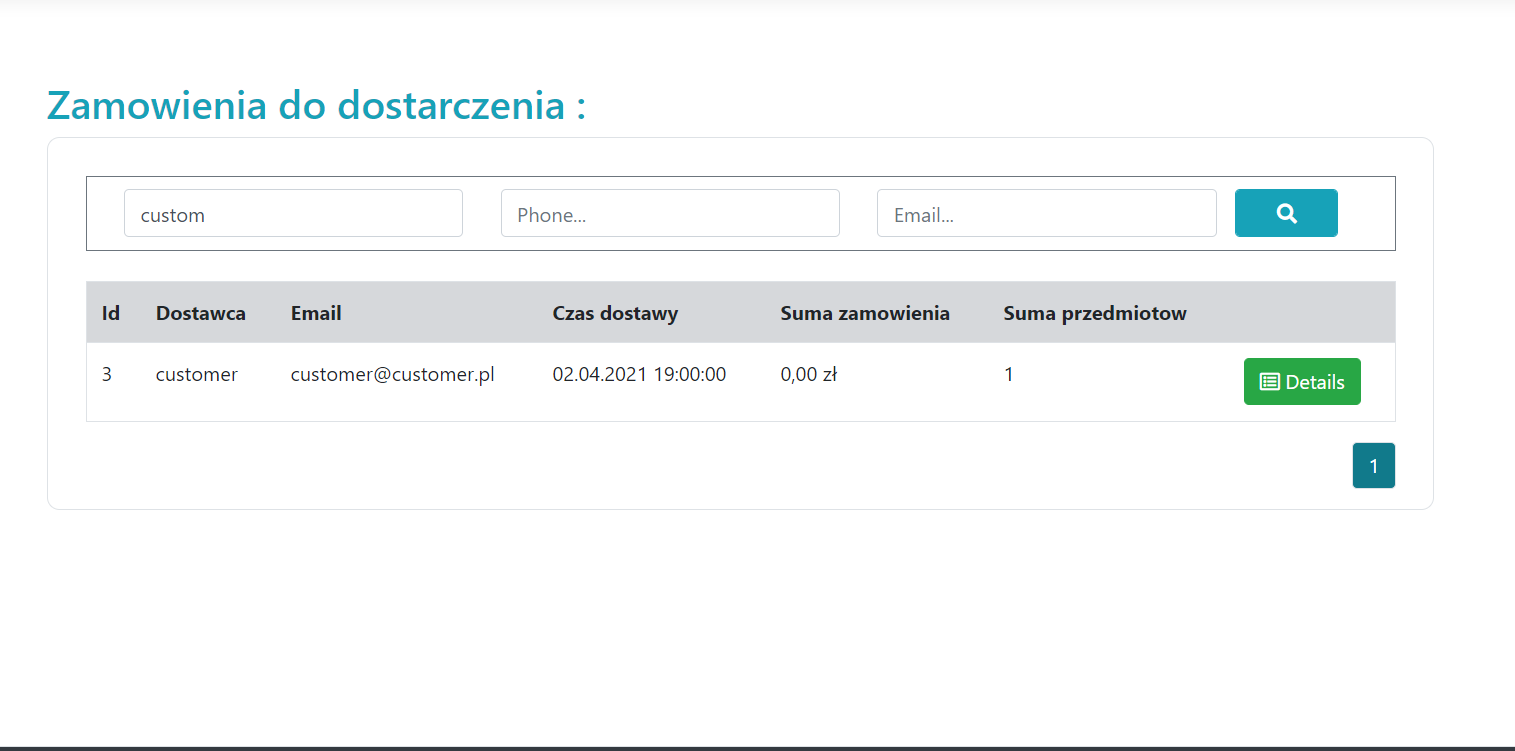
Efektem testu powinien być wynik wyrazony w rpm requests per minut.

Testy funkcjonalne – Michał Baca

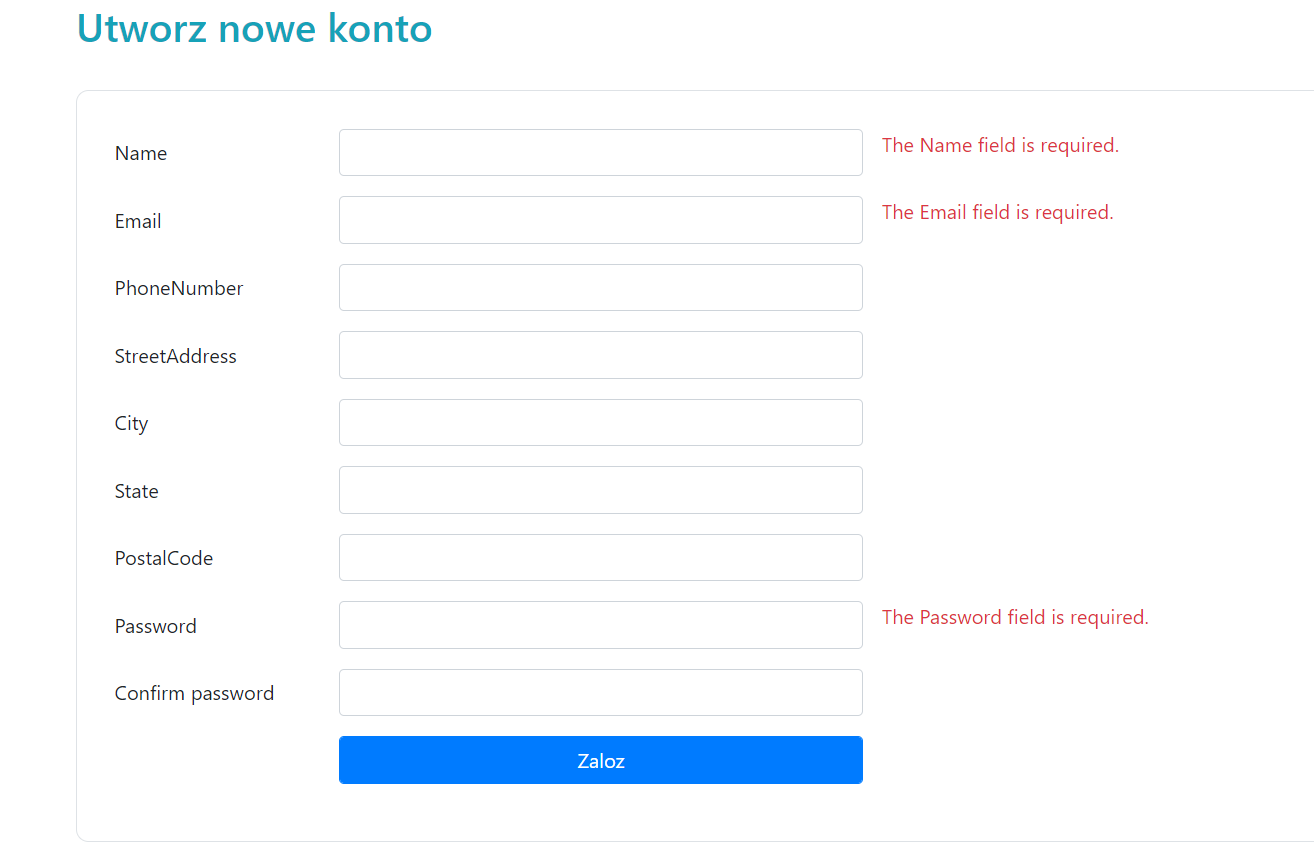
Testy funkcjonalne postanowiono przeprowadzić poprzez wykonanie wybranych akcji w aplikacji.

Podczas wyszukiwania zleceń kurier może wyszukać według nazwy. Poniższe zrzuty ekranu obrazują działanie programu.

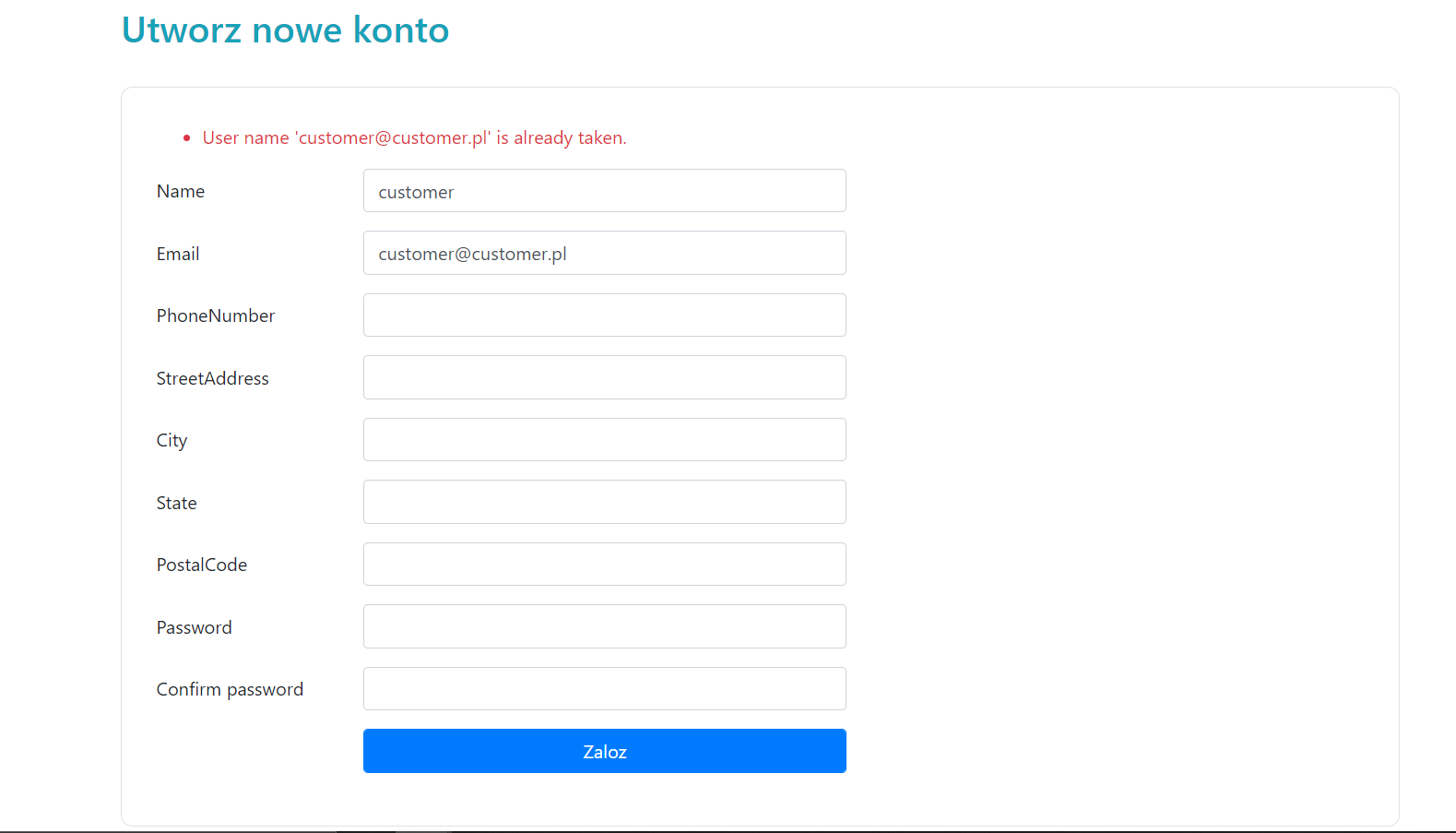




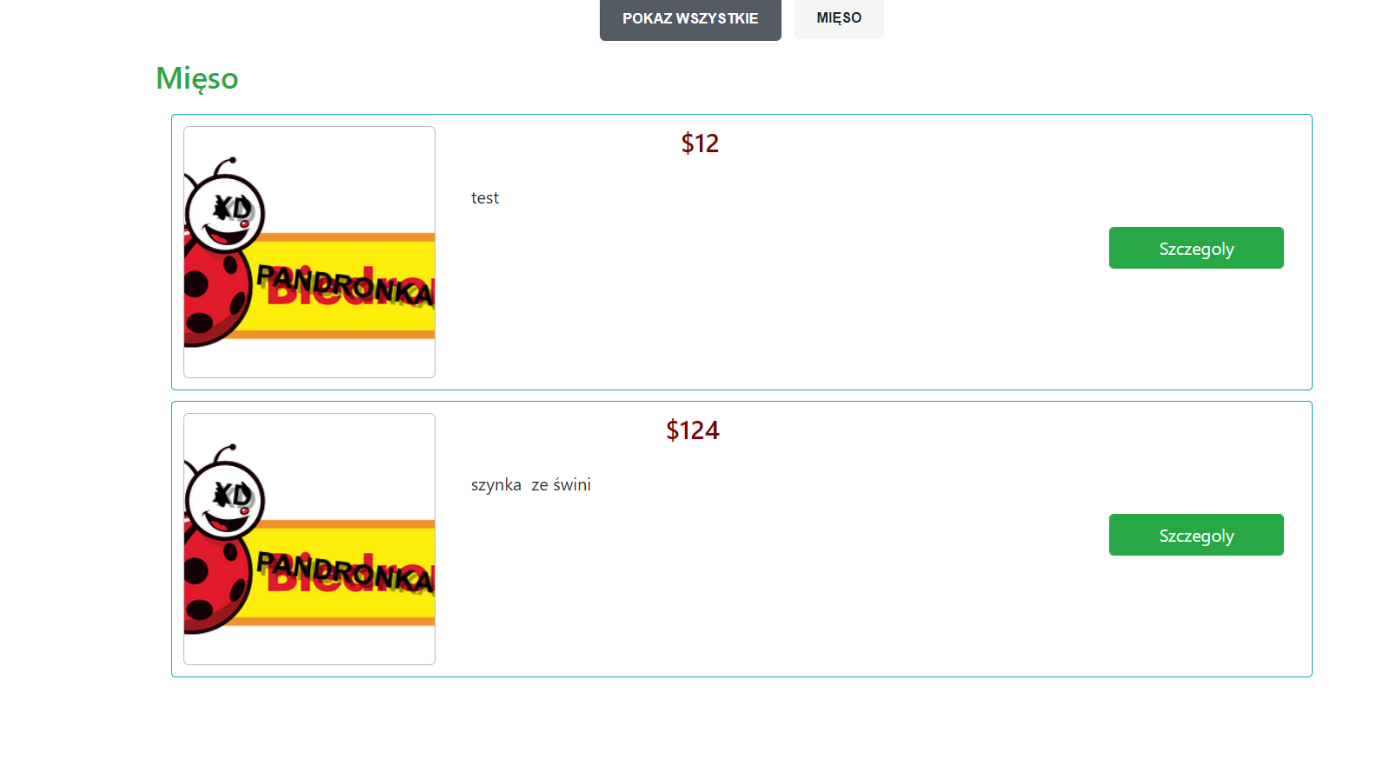
Podczas rejestracji użytkownika system wymaga wpisania wszystkich niezbędnych danych.



System nie pozwala na dodanie istniejącego już adresu mailowego, który jest polem unikatowym w bazie.



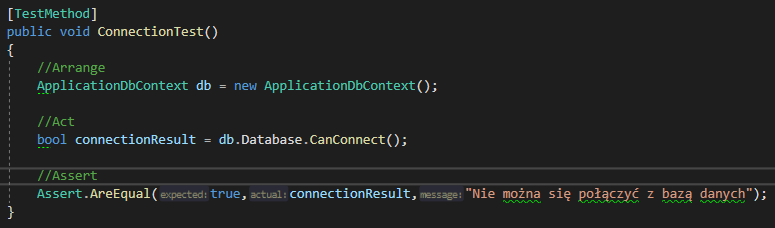
Jednym z założonych testów również było wykonanie sprawdzenia dodawania produktów. Weryfikacji podlegało prawidłowe wyświetlanie dodanego produktu na stronie głównej. Produkty dodają się prawidłowo jeden pod drugim z takim samym odstępem graficznym.



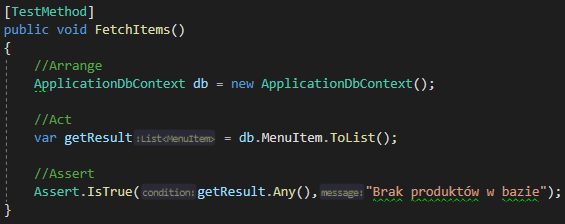
Testy jednostkowe – Mateusz Rzewnicki

Testy Bazy

**Test połączenia z bazą danych**

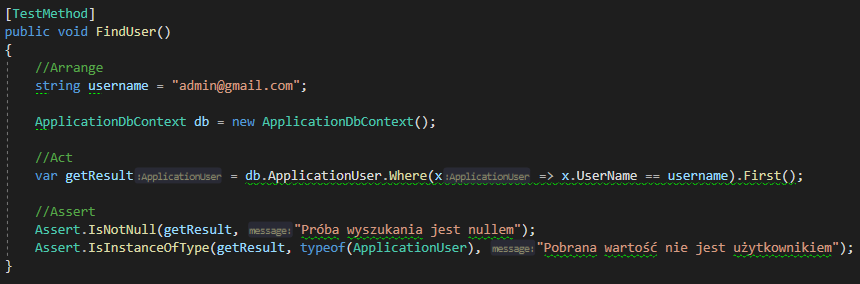


**Test pobrania danych**



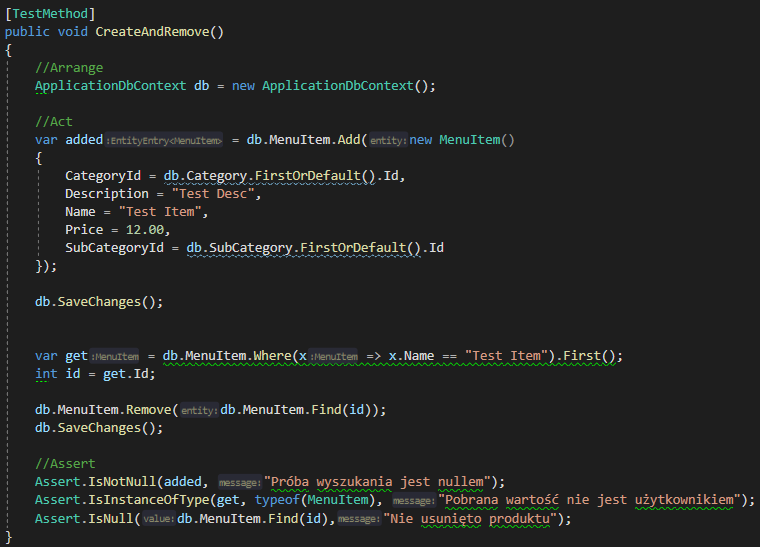
Testy Użytkowników

**Test odnalezienie użytkownika według loginu**



Testy Produktów

**Test utworzenia i usunięcia produktu testowego**



Rezultaty testów

