Co powinna zawierać dokumentacja:

1. Dokumentacja zadania "[Diagram przypadków użycia](https://moodle.wsiz.edu.pl/mod/assign/view.php?id=41566)"

2. Dokumentacja do zadania "[Diagram klas](https://moodle.wsiz.edu.pl/mod/assign/view.php?id=43459)"

3. Dokumentacja zadania "[Diagram stanów](https://moodle.wsiz.edu.pl/mod/assign/view.php?id=45090)"

4. Dokumentacja zadania "[Paymentprocessor diagram](https://moodle.wsiz.edu.pl/mod/assign/view.php?id=47946" \o "Paymentprocessor diagram)"

5. Dokumentacja zadania "Migracje, encje api"

UWAGI:

Link do github'a. Jeżeli zadanie było oddane mailowo, proszę wskazać w dokumentacji.

Spis treści

[Wstęp 2](#_Toc136861317)

[1. Diagram przypadków użycia 3](#_Toc136861318)

[2. Diagram klas 4](#_Toc136861319)

[3. Diagram stanów 5](#_Toc136861320)

[4. Paymentprocessor diagram 6](#_Toc136861321)

[5. Migracje, encje api 7](#_Toc136861322)

[Wniosek: 8](#_Toc136861323)

**Wstęp**

Link do github głównego: <https://github.com/Tuxuy3Dom/Inzynieria_Oprogramowania/tree/master>

Posiada dwie gałęzie: master – zawiera laboratorium pierwszy oraz laboratorium związany z pracą Angulara; LAB3 – w tej gałęzi zawiera się większość wykonanych prac laboratoryjnych, link do tej gałęzi: <https://github.com/Tuxuy3Dom/Inzynieria_Oprogramowania/tree/LAB3>

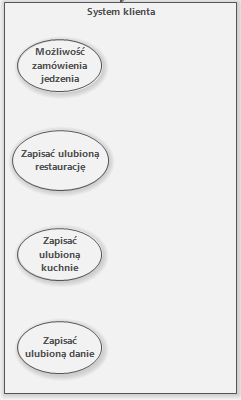
Na pierwszym zajęciu opracowywaliśmy metody budowy testów modułowych z wykorzystaniem dedykowanych narzędzi testowania jednostkowego. Czyli proces inspekcji kodu źródłowego. Pracowaliśmy w środowisku IDE: Visual Studio. Do tego mieliśmy zainstalowane: .NET SDK i do tego trzeba jeszcze było zainstalować: nuget XUnit, Bogus**.**

Dalej na zajęciach laboratoryjnych mieliśmy styczność z takimi metodami tworzenia diagramów jak: Diagram przypadków użycia, Diagram klas, Diagram stanów oraz paymentprocessor diagram. Wykonanie danych diagramów robiliśmy w programie: Software Ideas Modeler – narzędzie case do tworzenia diagramów, projektowania i analizy oprogramowania.

Także pracowaliśmy z frameworkiem ABP, który pracuje z Angular framework, oraz SQL Server.

1. **Diagram przypadków użycia**

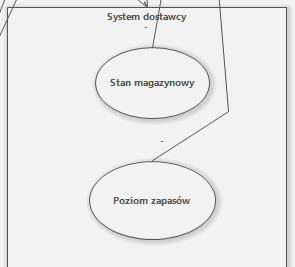
Zaczynałem od tworzenia Systemu zarządzania zamówieniami, do której dodałem systemy. W których przypisywałem funkcjonalności które będzie zawierać nasza aplikacja. Mam cztery systemy dla czterech różnych role: System klienta, System restauratora, System dostawcy i System sanepid.



Rysunek 1 System klienta

Źródło: Software Ideas Modeler program

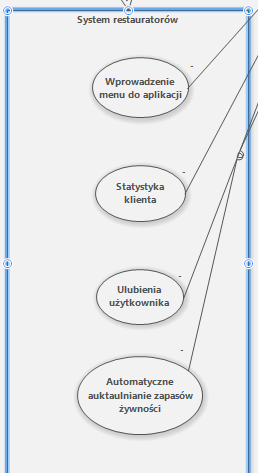
System klienta zawiera takie możliwości jak: możliwość zamówienia jedzenia – czyli użytkownik, który wchodzi do aplikacji ma możliwość złożyć zamówienie na dostawę jedzenia; także może zapisać restaurację którą polubił i mieć szybki dostęp do niej; są też opcja zapisania ulubionej kuchnie oraz danie.



Rysunek 2 System dostawcy

Źródło: Software Ideas Modeler program

System dostawcy zawiera tylko dwie możliwości stan magazynowy – gdzie pokazane ile jeszcze zostało produktów na stanie oraz poziom zapasów – czyli informuje dostawce na przykład o to że jakiś tam produkt zostaw już całkiem sprzedany albo przekroczył stan minimalny który mówi że potrzebnie robić już dostawę konkretnego produktu.

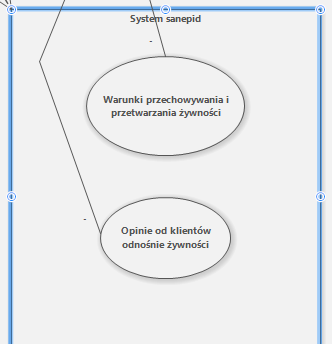


Rysunek 3 System restauratorów

Źródło: Software Ideas Modeler program

System restauratorów posiada możliwości wprowadzenia menu do aplikacji oraz w tej opcji jest możliwość też modyfikowania wprowadzonego menu; statystyka klienta mówi o tym co najbardziej lubią w danym miejscu; ulubienia pokazuje wszystkie polubienia które robią użytkownicy w danej restauracji; automatyczne uaktualnianie zapasów żywności wylicza w systemie ilość wydanych zamówień oraz ile jeszcze jest możliwe do zrealizowania według aktualnych danych zapasowych.

System sanepidowi informuje sanepid o tym jakie warunki przechowywania i przetwarzania żywności w konkretnej restauracji; także dostępna możliwość dostania od klientów zgłoszeń oraz opinii od klientów odnośnie żywności w lokalach.

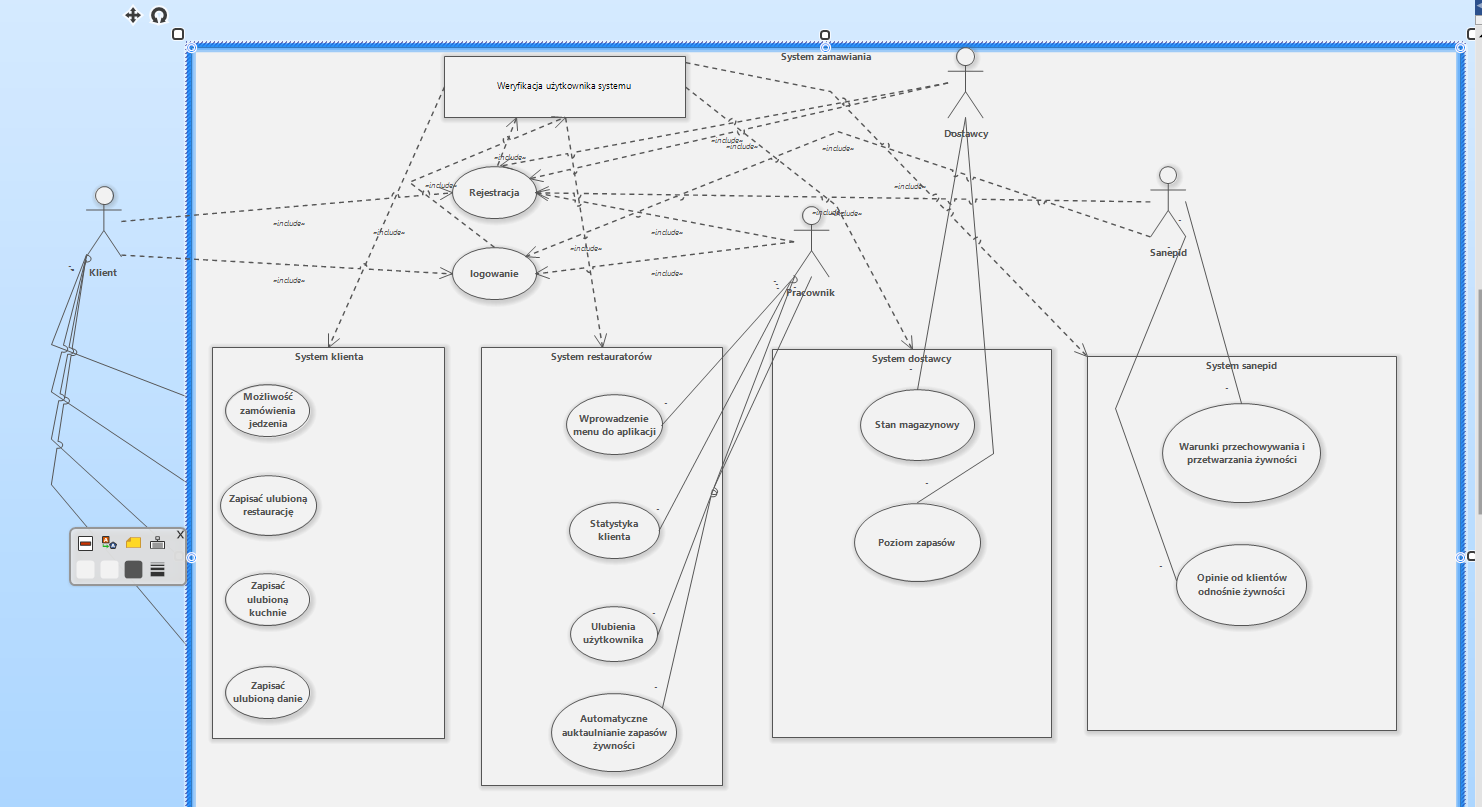


Rysunek 4 System sanepidowy

Źródło: Software Ideas Modeler program

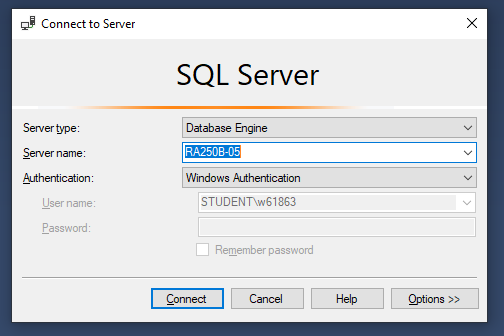
Przed dostępem do danych systemów, które wymieniłem wyżej, każdy użytkownik zaczyna od logowania się do systemu, w procesie logowania przechodzi weryfikacja danego użytkownika na to jaki system jemu udostępnić i już od rolę użytkownika. Która jest przepisana wyświetla się odpowiedni system zarządzania aplikacją.

Poniżej przedstawiam cały system zarządzania zamówieniami.

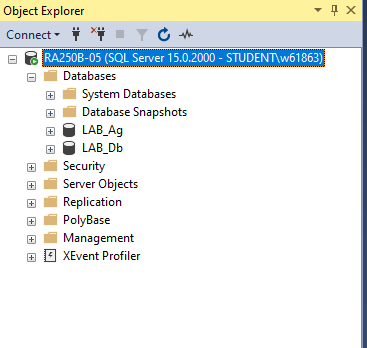


1. **Diagram klas**
2. **Diagram stanów**
3. **Paymentprocessor diagram**
4. **Migracje, encje api**

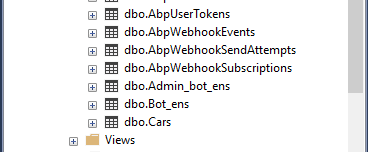
Tworzenie migracji robiliśmy w programie Visual Studio 2019. Oraz dodawaliśmy to do bazy danych Microsoft SQL Server Management Studio.



W databases stworzyłem bazę z nazwą: LAB\_Ag

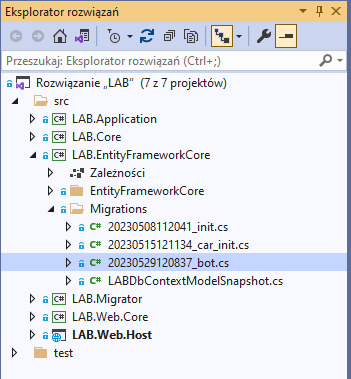


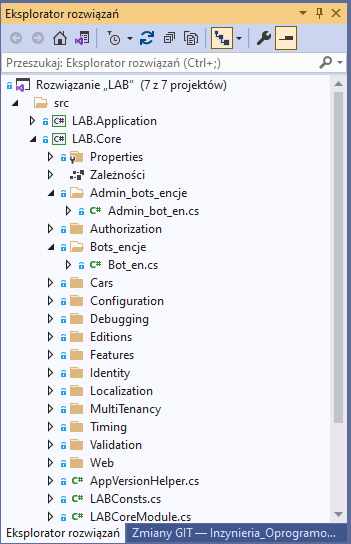
I do tej bazy do sekcji tablic robiłem migracje moich encji: pod nazwą dbo.Admin\_bot\_ens oraz dbo.Bot\_ens



Robiliśmy ustalenie: LAB.Web.Host – to pozwoli nam startować nasz projekt

Migracje robiliśmy w Konsoli menedżerów pakietów: w LAB.EntityFrameworkCore **–** za pomocą polecenia: add-migration i żeby było widoczne w naszej bazie to piszemy polecenie: update-database.





**Wniosek:**