

Politechnika Warszawska

WYDZIAŁ MECHANICZNY TECHNOLOGICZNY



# Praca dyplomowa inżynierska

na kierunku Automatyzacja i Robotyzacja Procesów Produkcyjnych

Projekt systemu zamówień do przedsiębiorstwa  
gastronomicznego

numer pracy według wydziałowej ewidencji prac:

**Konstantin Karavaev**

numer albumu 303144

promotor

dr inż. Ewa Bednarczyk

konsultacje

dr inż. Ewa Bednarczyk

WARSZAWA 2023



KARTA REJESTRACJI TEMATU PRACY		
Praca dyplomowa inżynierska		
Instytut Mechaniki i Poligrafii		Zakład Konstrukcji Maszyn i Inżynierii Biomedycznej
<b>Student</b> (imię i nazwisko)	Konstantin Karavaev	
Numer albumu	303144	Kierunek studiów Automatyzacja i Robotyzacja Procesów Produkcyjnych
Specjalność		Rodzaj studiów Stacjonarne I stopnia
Tytuł pracy (w języku studiów) <i>Projekt systemu zamówień do przedsiębiorstwa gastronomicznego</i>		
Tytuł pracy (w drugim języku) Design of an ordering system for a catering company		
Opis i cel pracy Podstawowym celem pracy jest zaprojektowanie systemu automatyzacji zamówień gastronomicznych, który będzie spełniać zdefiniowane wcześniej potrzeby personelu gastronomicznego.		
Zakres pracy Praca obejmować będzie przegląd warunków ergonomicznych pracy i potrzeb personelu w wybranym sektorze gastronomii oraz przegląd potrzeb biznesowych wybranego przedsiębiorstwa, zdefiniowanie potrzeby i problemu do rozwiązania, zaprojektowanie oraz przygotowanie aplikacji do zamówień gastronomicznych, zaprojektowanie oraz przygotowanie serwera na potrzeby stosowania aplikacji w przedsiębiorstwie, zaprojektowanie i wytworzenie metodą przyrostową wielofunkcyjnego uchwytu na urządzenie obsługujące aplikację, krytyczną analizę całości zaprojektowanego systemu zamówień.		
Planowany termin zakończenia pracy: 24.01.2023		
Promotor/Opiekun (stopień, imię i nazwisko)	dr inż. Ewa Bednarczyk	
Potwierdzenie przyjęcia pracy do wykonania przez studenta	Potwierdzenie objęcia opieki nad pracą przez promotora/opiekuna	
Data i podpis	Data i podpis	
Opis postępów w realizacji pracy (opcjonalny) – wypełnia promotor/opiekun		
Numer rejestracyjny pracy w APD	Potwierdzenie przez promotora/opiekuna pozytywnej oceny pracy i jej gotowości do obrony	
Data i podpis osoby rejestrującej	Data i podpis	
Ocena (dotyczy pracy przejściowej)	Akceptacja tematu przez Kierownika Zakładu	
Data i podpis opiekuna pracy	Data i podpis	

## Obieg dokumentu:

1. Student po uzgodnieniu z promotorem/opiekunem tytułu i zakresu pracy wypełnia formularz FOR-APD, drukuje go w 2 egzemplarzach i podpisuje.
2. Promotor/opiekun potwierdza tytuł pracy, wyraża zgodę na jej prowadzenie i uzyskuje akceptację kierownika zakładu.
3. Promotor/opiekun niezwłocznie przekazuje formularz w wersji elektronicznej (skan) do sekretariatu Instytutu w celu rejestracji w APD.
4. Niniejszą kartę należy dołączyć do pracy (również w wersji elektronicznej w systemie APD).



## **Streszczenie**

### **Projekt systemu zamówień do przedsiębiorstwa gastronomicznego**

Głównym zagadnieniem pracy jest opracowanie systemu automatyzacji przyjęcia zamówień gastronomicznych skierowany na zaspokojenie niektórych potrzeb mikro oraz małych przedsiębiorstw. Potrzeby te zostały wykryte podczas przeglądu warunków funkcjonowania realnego biznesu na przykładzie konkretnej pizzerii.

Opracowany system składa się z tabletu opartego na systemie Android, drukarki termicznej dla paragonów, obudowy w głównej mierze utworzonej metodą przyrostową druku 3D oraz serwera z bazą danych dań dostępnych w restauracji. Proces projektowania architektury systemu również został przedstawiony w pracy. Cały system został napisany w języku Pythone oraz jego Frameworkach (Python, Python KivyMD, Python Django).

**Słowa kluczowe:** Python, KivyMD, Django, architektura systemu IT, potrzeby przedsiębiorstwa gastronomicznego.

## **Abstract**

### **Project of an order system for a catering company**

The main topic of the work is to develop an automation system for gastronomic orders aimed at satisfying certain needs of micro and small businesses. These needs were identified during the review of the operating conditions of the real business using a specific of a real pizzeria.

The developed system consists of a tablet on the Android system, a thermal printer for receipts, a box mainly created by the incremental method of 3D printing and a server with a database of dishes available in the restaurant. The process of designing the system architecture is also presented in the work. The whole system was written in the Python language and its Frameworks (Python, Python KivyMD, Python Django).

**Keywords:** Python, KivyMD, Django, IT system architecture, needs of gastronomical businesses.

## Spis treści

1.	Wstęp.....	8
1.1.	Wprowadzenie.....	8
1.1.1.	Wstęp.....	8
1.1.2.	Przegląd przedsiębiorstwa.....	8
1.1.4.	Wybór zagadnienia do optymalizacji.....	10
1.2.	Cel pracy.....	11
2.	Istniejące rozwiązania.....	12
3.	Opis sprzętu/architektury systemu.....	15
3.1	Dobór sprzętu.....	15
3.2	Opracowanie architektury systemu.....	21
4.	Server KusTech.....	24
4.1	Wymagania.....	24
4.2	Technologia.....	24
4.3	Proces projektowy.....	27
4.4	Serwer.....	34
5.	Aplikacja KusTech.....	37
5.1	Wymagania.....	37
5.2	Technologia.....	37
5.3	Proces projektowy.....	38
5.4	Instalacja.....	46
6.	Obudowa.....	49
6.1	Wymagania.....	49
6.2	Technologia.....	49
6.3	Proces projektowy.....	51
7.	Testowanie.....	55
8.	Wnioski i podsumowanie.....	58
9.	Bibliografia.....	60
10.	Spis rysunków.....	63
11.	Spis tabel.....	66
12	Załączniki.....	67
	Załącznik 1, Plik „models.py” aplikacji webowej.....	67
	Załącznik 2, Funkcją odpowiadającą za pobieranie danych z serwera.....	69
	Załącznik 3, Funkcja odpowiadająca za wydruk paragonu.....	70

# **1. Wstęp**

## **1.1. Wprowadzenie**

### **1.1.1. Wstęp**

Niniejsza praca inżynierska jest poświęcona zademonstrowaniu procesu opracowania systemu cyfrowego wysokotechnologicznego mającego na celu zaspokojenie wcześniej wykrytych potrzeb małych przedsiębiorstw gastronomicznych.

Wdrażanie systemów cyfrowych przez małe firmy pozwala na zwiększenie ich konkurencyjności poprzez automatyzację pewnych działań mających charakter cykliczny i monotony. Systemy IT pozwalają również na wyeliminowanie człowieka z niektórych procesów biznesowych, co skutkuje zmniejszeniem możliwości wystąpienia błędu ludzkiego oraz zmniejszeniem potencjalnych wydatków właściciela na zespół pracowniczy. Poza tym pozytywnie wpływają na pracę personelu – wykonuje niektóre ich obowiązki, co doprowadza do zwiększenia wolnego czasu oraz wydłużeniem przerw w miejscu pracy.

Powyższe argumenty coraz częściej skłaniają potencjalnych przedsiębiorców na inwestowanie w różne systemy wysokotechnologiczne, zwłaszcza warunkach wysokiej konkurencji rynkowej, w których żyjemy dziś. Gdy przedsiębiorca już podjął decyzję o digitalizacji swojego biznesu, pojawia się następujące pytanie – komu zapłacić, aby biznes był sprawniejszy? Czy wybrać jedno z dużej liczby gotowych rozwiązań i przebudować pod niego swoje procesy biznesowe, czy jednak napisać swój własny system cyfrowy, który by idealnie pasował do już działającego modelu biznesowego? Pierwsza opcja jest zdecydowanie tańsza, ale mająca większe ryzyko dla biznesu w związku z przebudową procesów wewnętrznych. Natomiast druga opcja jest bezpieczniejsza, ponieważ przed przystąpieniem do pisania systemu programista analizuje rzeczywisty model firmy, jego słabe i mocne strony, w wyniku czego jego wdrożenie do modelu biznesowego przechodzi szybko i praktycznie bez zatrzymania procesów wewnętrznych. Również przy stosowaniu drugiej opcji istnieje możliwość połączenia nowego systemu z już działającymi systemami w biznesie poprzez dodatkowe napisanie kodu dla komunikacji między nimi.

Niniejsza praca skupia się na drugiej opcji - napisanie własnego systemu cyfrowego dla już działającego modelu biznesowego.

### **1.1.2. Przegląd przedsiębiorstwa**



Jako przykład małego przedsiębiorstwa gastronomicznego będziemy rozpatrywali pewną działającą pizzerię, znajdującą się w Nowym Sączu. Jest to niewielka firma mająca ledwie 11 pracowników. W skład personelu wchodzi pięciu kucharzy, administrator, dwójka kasjerów, dyrektor oraz dwie osoby pomagające na całym obiekcie.

Zadaniem kucharzy jest opracowanie i realizacja otrzymanych od kasjerów zamówień oraz dbanie o porządek w kuchni.

Kasjerzy odpowiadają za przyjęcie zamówień od klientów, ich rozliczenie oraz przekazanie informacji do kucharzy.

Osoby pomagające na obiekcie dbają o porządek w głównej jadalni, sprzątają w razie potrzeby oraz realizują inne funkcje, które mogą być przydatne dla normalnego funkcjonowania procesów biznesowych.

Administrator natomiast zajmuje się wszystkimi rzeczami logistycznymi, obsługą skarg klientów oraz innymi formalnościami, występującymi podczas pracy pizzerii.

Dyrektor kieruje załogą, rozlicza kwestie finansowe oraz odpowiada przed urzędem skarbowym. Również dyrektor wybiera dalszy kierunek rozwoju firmy.

Pięciu kucharzy spędza większość swojego czasu roboczego w kuchni, gdzie powinni pracować w warunkach zapewniających im bezpieczeństwo. Odpowiednie ustawienie narzędzi kuchennych takich, jak na przykład noży, garnków oraz patelni, może pomóc w zmniejszeniu ryzyka wypadków na stanowisku roboczym. Odpowiednie oświetlenie i wentylacja są również ważne w rozpatrywanej sytuacji, aby zapewnić komfort i bezpieczeństwo w trakcie pracy.

Administrator placówki oraz dwóch kasjerów spędzają dużą część swojego czasu roboczego w obszarze stanowiska z kasą. Ergonomiczne stanowisko pracy z wygodnym krzesłem i blatem do wykonywania swoich obowiązków pozwala im pracować bez nadmiernego napięcia mięśni i bólu pleców.

Dyrektor oraz dwójka pracowników pomagających na całym obiekcie muszą poruszać się po całej pizzerii oraz wykonywać różne zadania bieżące takie, jak sprzątanie, serwowanie klientom i inne czynności wymagane dla dobrego funkcjonowania placówki. Dlatego ich stanowiska powinny być wyposażone w urządzenia mobilne takie, jak wózki lub kosze. Ma to na celu zminimalizować ryzyko urazu oraz zmęczenia mięśni pracowników.

Model biznesowy oraz role zespołu pracowników dobrane są we właściwy sposób, ponieważ przychody firmy są wyższe od wydatków oraz firma dość stabilnie rozwija się na Polskim rynku gastronomicznym.

### **1.1.3. Wykrycie nieoptymalnych rozwiązań**

Cały proces złożenia oraz realizacji zamówienia w rozpatrywanej restauracji dla klienta wygląda w następujący sposób:

1. U klienta pojawia się potrzeba i nastrój zainwestować środki pieniężne w jedzenie w rozpatrywanej pizzerii.
2. Klient przychodzi do pizzerii i wybiera artykuły spożywcze.
3. Czeką w kolejce przy kasie.
4. Przekazuje informację o wybranych produktach kasjerowi.
5. Realizuje opłatę za wybrane produkty kartą lub gotówką.
6. Kasjer przekazuje na kuchnię informację o zamówieniu.
7. Kuchnia realizuje zamówienie i przekazuje gotowe danie kasjerowi.
8. Kasjer przekazuje zamówienie klientowi.

Ze strony restauracji natomiast proces złożenia i realizacji zamówienia wygląda nieco inaczej:

1. Kasjer oczekuje na przybycie klienta.
2. Kasjer rozpoczyna obsługę klienta
3. Kasjer przetwarza otrzymaną informację i obsługuje płatność klienta.
4. Po otrzymaniu płatności przekazuje zamówienie na kuchnię oraz wprowadza go do wewnętrznej bazy danych.
5. Gdy kuchnia skończy z realizacją – powiadamia o tym kasjera oraz odznacza w bazie, że zamówienie jest gotowe przez kuchnię.
6. Kasjer woła klienta i przekazuje mu przygotowane zamówienie, po czym dodaje wpis do bazy, że zamówienie jest w pełni zrealizowane.

Po analizie powyższych procesów realizacji zamówień oraz biorąc pod uwagę fakt, że im więcej czynności w procesie biznesowym wykonywane są automatycznie, tym przedsiębiorstwo funkcjonuje szybciej oraz przynosi większe zyski, zostały wykryte następujące słabe miejsca:

1. Podczas dużego ruchu klient jest zmuszony na oczekiwanie w kolejce, co może go doprowadzić do długiego czasu oczekiwania, co negatywnie wpływa na doświadczenie klienta oraz na ogólny wizerunek całej placówki.
2. Po wyjściu gotowego dania z kuchni kasjer jest zmuszony do ustnego wołania klienta, co doprowadza czasami do nieporozumień i opóźnień w odbiorze zamówienia.
3. Trudności w możliwej personalizacji zamówienia przez klienta. Brak przedstawienia całej oferty składników oraz ograniczania czasowe ze względu na ruch klientów.
4. Brak systemu do monitorowania gotowości zamówienia klienta, co może doprowadzić do utraty lojalności oraz niezadowolenia klienta placówką.

Na próbie rozstrzygnięcia jednego z wymienionych problemów będzie oparta niniejsza praca.

#### **1.1.4. Wybór zagadnienia do optymalizacji**

Jedną z opcji rozstrzygnięcia pierwszego problemu jest zaprojektowanie oraz integracja do procesu realizacji zamówień informatycznego systemu automatyzacji przyjęcia zamówień gastronomicznych. Polega on na postawieniu obok kasjera wyświetlacza dotykowego z możliwością samodzielnego złożenia zamówienia. Integracja wymienionego rozwiązania

daje klientowi czekającemu w kolejce alternatywną metodę złożenia zamówienia, która doprowadza do szybszej jego obsługi oraz zmniejszenia do niezbędnego minimum kontaktu z zespołem pracowniczym w placówce.

Rozwiązanie drugiego problemu może być zrealizowane dzięki stosowaniu specjalistycznych „Przywołaczy klientów”. Są to urządzenia z bluetoothem, które wydawane są klientowi po złożeniu zamówienia. Gdy kuchnia przekazuje gotowe danie kasjerowi, kasjer aktywuje sygnał wibracyjny na wydany „Przywołacz”, co sygnalizuje klientowi, że należy odebrać gotowe danie.

Trzeci problem można rozwiązać, zwiększając możliwą ilość czasu znajdowania się przy kasie oraz demonstracją oferty składników tuż w miejscu składania zamówień. Limit czasowy przy tej samej przepustowości możemy zwiększyć, zwiększając ilość kas. Rozstrzygnięcie pierwszego oraz trzeciego problemu może być zrealizowane w jednym systemie automatyzacji.

Do likwidacji czwartego problemu należy użyć wyświetlacza z demonstrowaniem numerów zamówień, rozmieszczonym przy kasie lub w pomieszczeniu ze stolikami. Po złożeniu zamówienia jego numer trafiłby do kolumny z zamówieniami w trakcie realizacji, natomiast po jego przygotowaniu przez kuchnię – migrowałby do kolumny z gotowymi daniami.

W tej pracy postanowiono skupić się na rozwiązaniu pierwszego problemu ze względu na jego pracochłonność oraz na wymagany wysiłek intelektualny.

Wdrażanie takiego systemu powinno pozytywnie wpłynąć na ergonomię pracy części pracowników – część klientów wybierze korzystanie ze stanowiska samoobsługowego zamiast użycia tradycyjnej kasy. Spowoduje to potencjalne zwiększenie czasu wolnego kasjerom.

## **1.2. Cel pracy**

Celem pracy jest zaprojektowanie architektury systemu automatyzacji przyjęcia zamówień gastronomicznych, zaprojektowanie oraz napisanie serwera z bazą danych dań rozpatrywanej restauracji, zaprojektowanie aplikacji na Android dla obsługi klienta, zaprojektowanie obudowy oraz jej wykonanie metodą przyrostową. Po wykonaniu powyższych punktów będą przeprowadzone testy spójności wszystkich elementów oraz będzie przeprowadzona analiza wykonanej pracy.

## 2. Istniejące rozwiązania

Na każdym rynku są konkurenci – w naszym przypadku ta zasada nie została złamana. Najslawniejszym w naszym przypadku jest każdemu dobrze znana międzynarodowa korporacja „McDonald’s Corporation” działająca w branży fast food, która jako pierwsza na rynku gastronomicznym wprowadziła do swoich restauracji automatyczne stanowiska do samoobsługi. Automaty do samoobsługi są elektronicznymi urządzeniami, które pozwalają klientom na składanie zamówień oraz ich opłacanie bez konieczności interakcji z zespołem pracowniczym. McDonald’s wprowadził takie automaty, mając na celu zwiększenie efektywności oraz wygodę obsługi klientów. Dokumentacja techniczna używanych automatów nie jest dostępna publicznie, natomiast na niektóre podobne urządzenia jest ona dostępna [1].

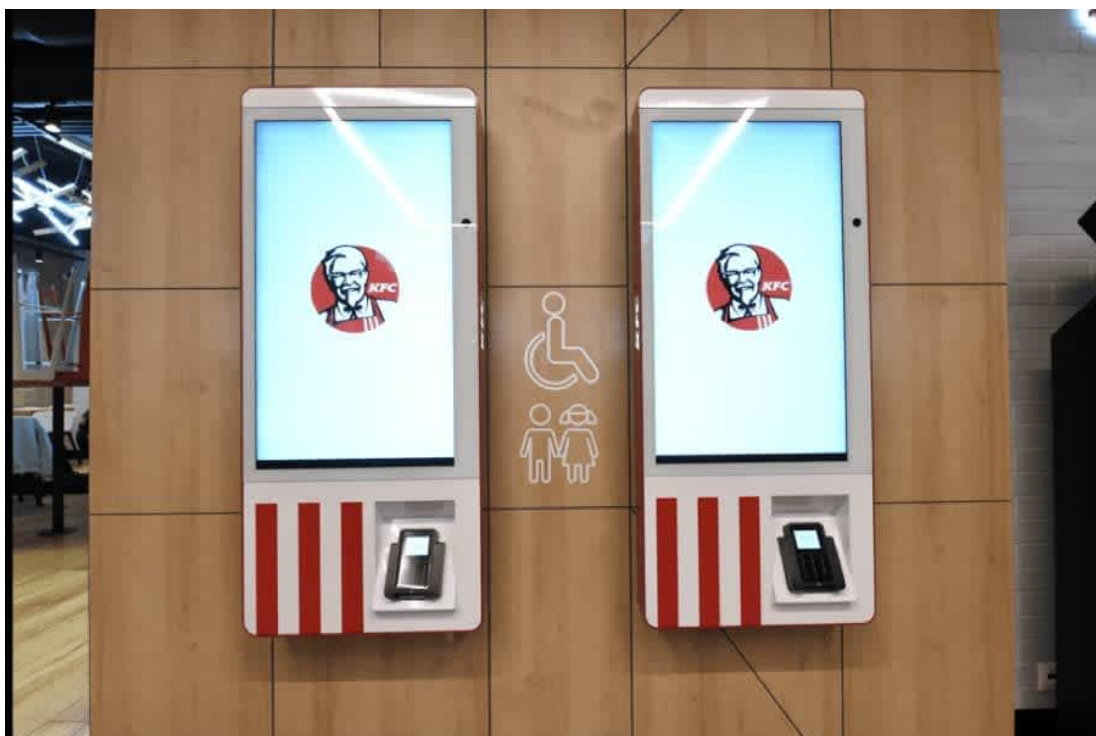


**Rysunek 1, Automaty do samoobsługi w McDolald`s.**

Źródło: Strona internetowa tapbox.eu : <https://tapbox.eu/more-money-is-spent-using-self-service-kiosks/>, data aktualizacji: 10.05.2023.

Automaty w McDonald's posiadają dotykowy wyświetlacz, korzystając z którego klienci mogą przeglądać ofertę, wybierać produkty, modyfikować składniki zamówienia oraz realizować płatność bezgotówkową (rysunek 1). Po dokonaniu zamówienia klient otrzymuje paragon ze specjalnym numerem, który pozwala na monitorowanie statusu zamówienia oraz upoważnia do odbioru gotowego posiłku. Automaty w McDonald's mają w sobie również opcję personalizacji zamówień, takie jak na przykład możliwość zmiany składników pewnych pozycji, wybór rodzaju chleba lub dodatków do burgerów.

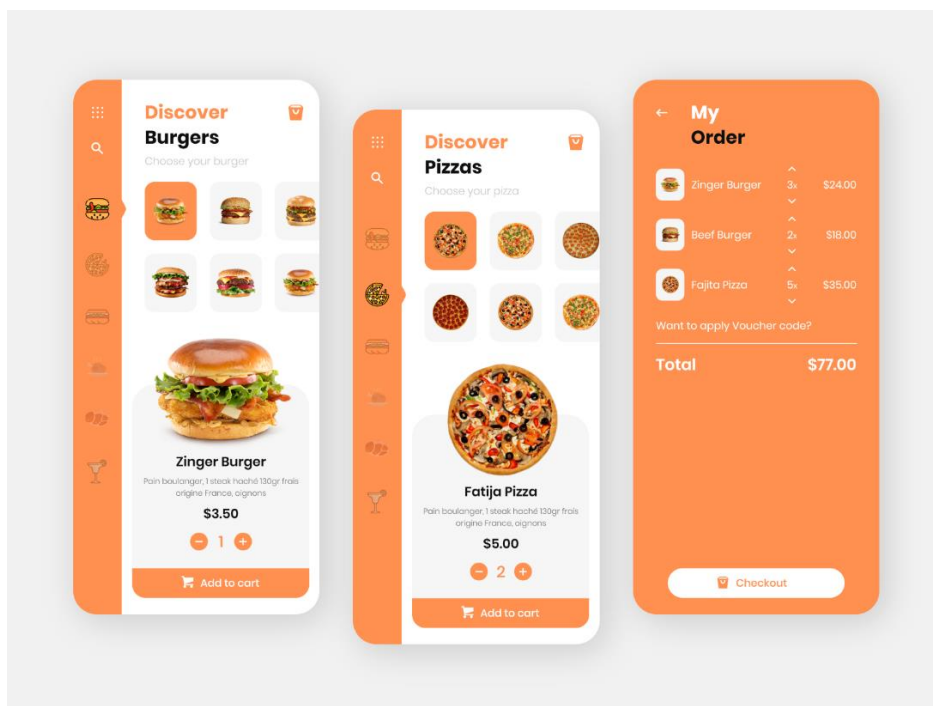
Poza McDonald's inne sieci też nie stoją w miejscu - konkurenci również wprowadzili do swoich placówek automaty do samoobsługi, które mają na celu zwiększyć efektywność obsługi klientów oraz zmniejszyć czas czekania w kolejce. Przykładowi konkurenci, którzy stosują podobne rozwiązania, to są na przykład KFC, Burger King czy nawet Ikea (rysunek 2). Automaty do samodzielnego składania zamówień w wyżej wymienionych sieciach oferują podobne funkcje, takie jak personalizacja zamówień, możliwość realizacji płatności kartą oraz śledzenie statusu zamówienia dzięki wydrukowanemu numerowi zamówienia na paragonie. W niektórych rozwiązaniach funkcjonalność różni się w niewielkim stopniu, ale główne ich zadania są niezmiennie – zmniejszenie kolejek oraz zwiększenie przepustowości kasjerów.



**Rysunek 2, Automaty do samoobsługi w KFC.**

Źródło: Strona internetowa kiosksoft.ru : <https://kiosksoft.ru/news/2020/06/11/kfc-zapuskaet-polnostyu-avtomatizirovannyj-restoran-v-moskve-68647> , data aktualizacji: 10.05.2023.

Rynek gastronomicznych aplikacji mobilnych również nie stoi w miejscu – ze względu na to, że napisana aplikacja będzie działała praktycznie na każdym urządzeniu Android/iOS, na które się je zainstaluje, próg wejścia jest bardzo niski – dlatego wiele twórców oprogramowania interesują się tym tematem. Natomiast za pomocą telefonu bez użycia specjalnych dodatkowych urządzeń nie ma możliwości wydrukowania paragonu lub pobrania płatności z karty (rysunek 3). Dlatego zazwyczaj same aplikacje nie znajdują szerokiego zastosowania w restauracjach.



**Rysunek 3, Przykład aplikacja do samoobsługi na smartfon.**

Źródło: Strona internetowa dribbble.com: <https://dribbble.com/shots/8724373-Fast-Food-App-Design> , data aktualizacji: 10.05.2023.

### 3. Opis sprzętu/architektury systemu

#### 3.1 Dobór sprzętu

Przed przystąpieniem do projektowania należy ustalić wytyczne, które będą wymagane w końcowym produkcie. Zostało ustalone, że należy stworzyć system automatyzacji przyjęcia zamówień gastronomicznych, ale na czym on dokładnie będzie polegał i co będzie mieć w środku?

Klient powinien przyjść do restauracji, zobaczyć kolejkę do kasjera oraz mój system samoobsługowy obok, podejść do niego, wybrać odpowiednie artykuły, złożyć zamówienie, zapłacić, otrzymać paragon z numerem zamówienia, po czym poczekać, póki ono nie będzie przygotowane, po czym odebrać i pójść degustować otrzymaną pizzę.

Przede wszystkim zacznę od tego, że dla jakiegokolwiek komunikacji z menu, klientowi należy go ładnie przedstawić. Można to zrobić, stosując duży monitor komputerowy lub telewizor. Sam monitor po podłączeniu do prądu pokazuje, że brakuje mu źródła sygnału – to źródło należy dołączyć. Źródłem jest urządzenie procesorowe, które jest w stanie przekazać obraz – do wymienionej roli idealnie pasuje komputer. Aby klient mógł wybrać cokolwiek z menu należy pozwolić mu wprowadzać dane do systemu z zewnątrz - trzeba dodać myszkę wraz z klawiaturą albo użyć monitora dotykowego. Dodanie myszki i klawiatury nie jest możliwe ze względu na estetykę – klient nie będzie chciał grzebać w komputerze podczas robienia zamówienia. Zostaje opcja z wyświetlaczem dotykowym. Zestaw z porządnego wyświetlacza dotykowego oraz odpowiedniej mocy komputera nie jest najtańszym rozwiązaniem na rynku – czy są jakieś alternatywy? Przeglądając oferty na współczesnym rynku, praktycznie każdy tablet ma w sobie dobrej jakości wyświetlacz dotykowy oraz odpowiednią moc obliczeniową, wystarczającą na potrzeby systemu [2]. Moc obliczeniowa tabletu pozwoli nie tylko na wyświetlenie menu, ale również na działanie specjalistycznej aplikacji, pozwalającej na złożenie zamówienia przez klienta.

Zostało ustalone, że będzie używany tablet – jaki i co powinien mieć w środku? Na chwilę obecną na rynku najbardziej rozpowszechnione są tablety z dwoma głównymi systemami operacyjnymi – IOS oraz Android. Urządzenia używające IOS są dobrej jakości, natomiast mają kilka potężnych wad, którymi między innymi są na przykład ich cena oraz bardzo mała utrudniona przestrzeń robocza dla developerów. Wyraża się to w braku możliwości instalacji aplikacji zewnętrznych, pobieranych z nieoficjalnych źródeł. Tablety używające systemu operacyjnego Android natomiast są bardziej przyjazne pod względem instalacji z nieoficjalnych źródeł oraz są zdecydowanie tańsze w porównaniu do produktów korporacji „Apple Inc” [2]. Podsumowując powyższe argumenty, wybieram tablet z systemem operacyjnym Android.

Jakie wymagania powinien spełniać tablet w projektowanym systemie? Nie każdy klient będzie miał dobry wzrok, dlatego wyświetlacz dotykowy nie może być mały. Najbardziej rozpowszechnione modele tabletów mają rozmiar wyświetlaczy w przedziale 7-11 cali, dlatego należy wybrać model najbardziej zbliżony do tej górnej granicy. Kwestia cenowa również odgrywa ogromną rolę przy wyborze – tworzymy sprzęt dla małych przedsiębiorstw, które nie będą miały kapitału na zakup drogich systemów informatycznych. Wybór



budżetowego tabletu również pozwoli na większe skalowanie systemu w przyszłości. Analizując polskie sklepy z AGD elektroniką, wybór padł na tablet „REALME Pad 10.4” 4/64 GB” – posiada on w miarę duży wyświetlacz oraz odpowiednią moc obliczeniową na potrzeby powstającego systemu[3] (rysunek 4).



**Rysunek 4, Tablet REALME Pad 10.4" 4/64 GB.**

Źródło: Strona internetowa mediamarkt.pl: [https://mediamarkt.pl/komputery-i-tablety/tablet-realme-pad-10-4-2022-wifi-4gb-64gb-szary?gclid=CjwKCAjwge2iBhBBEiwAfXDBRz9bcPuZTwxOn93wogCYAcnBo-m0ck\\_alELO\\_FgUtElgWX8asPjwERoCyjgQAvD\\_BwE&gclsrc=aw.ds](https://mediamarkt.pl/komputery-i-tablety/tablet-realme-pad-10-4-2022-wifi-4gb-64gb-szary?gclid=CjwKCAjwge2iBhBBEiwAfXDBRz9bcPuZTwxOn93wogCYAcnBo-m0ck_alELO_FgUtElgWX8asPjwERoCyjgQAvD_BwE&gclsrc=aw.ds) , data aktualizacji: 10.05.2023.

Po wybraniu odpowiednich artykułów spożywczych i ich opłaceniu klient powinien dostać potwierdzenie transakcji oraz numer swojego zamówienia. Poza wyświetleniem numerku na ekranie należy przekazać fizyczną jego kopię klientowi, żeby on mógł ją zademonstrować kasjerowi przy odbiorze zamówienia. Dobrym rozwiązaniem jest jego wydrukowanie na kartce, ponieważ poza samym numerkiem jeszcze musimy niektóre informacje klientowi przekazać. Zamiast zwykłej drukarki zalecane jest użycie drukarki termicznej ze względu na to, że nie potrzebują dodatkowego tuszu do pracy – tylko specjalny papier termiczny dla drukowania paragonów i wysoką temperaturę, którą zapewnia urządzenie drukujące [4] (rysunek 5).





**Rysunek 5, Papier termiczny firmy EMERSON.**

Źródło: Strona internetowa allegro.pl: <https://allegro.pl/oferta/rolka-kasowa-termiczna-57-20m-10szt-emerson-6711845882> , data aktualizacji: 10.05.2023.

Na rolę urządzenia drukującego należy wybrać dowolną drukarkę termiczną. Pytanie tylko jaki rodzaj komunikacji z urządzeniem wybrać? Na rynku są dostępne drukarki z podłączeniem przewodowym USB, bluetooth oraz podłączeniem stosującym protokół API poprzez kabel Ethernet. W moim projekcie bardzo ważnym czynnikiem jest stabilność funkcjonowania urządzenia, więc połączenie bezprzewodowe (bluetooth) odpada. Tablet nie będzie miał gniazda Ethernet, więc komunikacja poprzez API drukarki termicznej też będzie nieco utrudniona. Natomiast stosując odpowiednią przejściówkę z USB-C na USB, będę mógł podłączyć drukarkę bezpośrednio do tabletu. Do tego też będzie potrzebna przetwornica PL2303HX USB-TTL (rysunek 6) która przetwarza sygnał z komputera do formatu, który odpowiada drukarce, ale więcej szczegółów o podłączeniu elementów będzie w dalszej części pracy [5]. W naszym projekcie wybieram drukarkę GOOJPRT QR701 – jest to mała drukarka termiczna z przewodowym podłączeniem do USB [6] (rysunek 7). Stosowana dość często w projektach na platformie Arduino.



**Rysunek 6, Przetwornica PL2303HX USB-TTL.**

Źródło: Strona internetowa sklep.msalamon.pl: <https://sklep.msalamon.pl/produkt/konwerter-usb-rs232-pl2303-ttl/> , data aktualizacji: 10.05.2023.



**Rysunek 7, Drukarka termiczna GOOJPRT QR701.**

Źródło: Strona internetowa [pl.aliexpress.com](https://pl.aliexpress.com/item/4000089296652.html): <https://pl.aliexpress.com/item/4000089296652.html> , data aktualizacji: 10.05.2023.

Poza drukarką termiczną należy jeszcze podłączyć terminal płatniczy do pobierania opłat od klienta, natomiast w ramach tej pracy inżynierskiej nie jest to możliwe ze względu na to, że dla właściwej pracy z terminalem płatniczym należy zarejestrować go w systemie, co nie jest możliwe bez zarejestrowanej działalności gospodarczej. Nawet jeżeli działalność gospodarcza istnieje, to i tak by należało go zakupić oraz płacić każdego miesiąca pewną kwotę za jego utrzymanie (około 40 zł w zależności od producenta [7]), co też jest wydatkiem dla projektanta. Dlatego terminal będzie zastąpiony przyciskiem, symulującym sygnał, że płatność została zrealizowana.

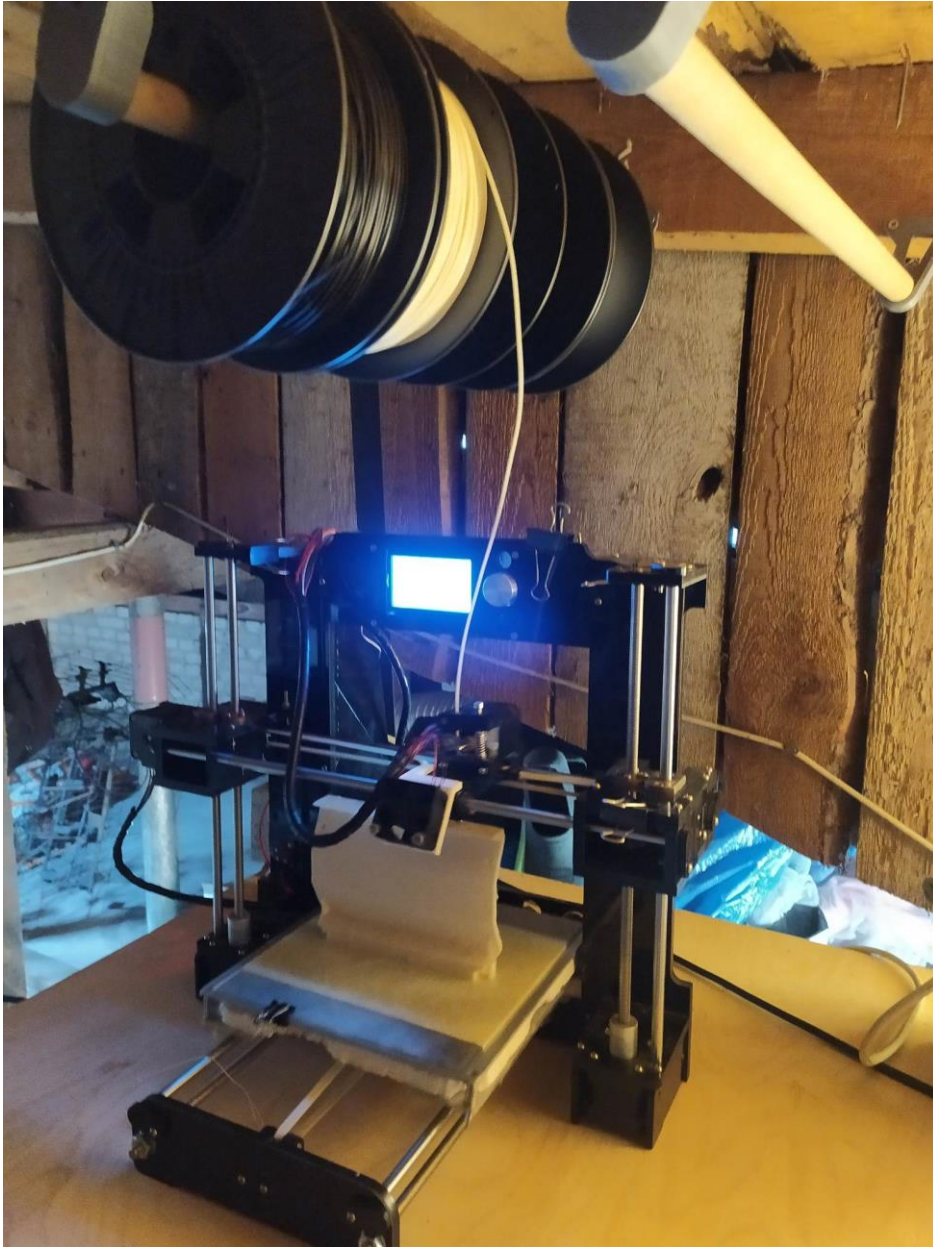
Tablet ma tylko jedno wejście USB-C, natomiast dla złączenia wszystkich elementów do jednego systemu to nie wystarczy. Będzie potrzebny adapter, mający co najmniej jedno wejście USB oraz jedno wejście USB-C dla podłączenia zasilania układu. Dla tego celu został wybrany „Adapter USB-C Tradebit 6318” na 5 wejść [8]. Posiada on dwa wejścia USB, 1 wejście USB-C, 1 wejście HDMI oraz jedno wejście do kabla Ethernet (rysunek 8).



**Rysunek 8, Adapter USB-C Tradebit 6318.**

Źródło: Strona internetowa ceneo.pl: <https://www.ceneo.pl/137069814> , data aktualizacji: 10.05.2023.

Po dokonanych wyborze fizycznych komponentów systemu należy się zastanowić nad obudową, w której wszystkie te elementy będą przechowywane. Są dwie znane projektantowi metody utworzenia korpusu – metoda wycinania frezarką CNC [9] oraz metoda przyrostowa na drukarce 3D [10] (rysunek 9). Każda z nich posiada swoje wady i zalety, natomiast w tym projekcie ze względu na złożoność kształtów korpusu oraz umiejętności projektowania konstruktora wybrana została metoda przyrostowa na drukarce 3D z użyciem tworzywa PLA. Więcej na temat korpusu będzie powiedziane w rozdziale 6 - „Obudowa”.



**Rysunek 9, Drukarka do druku 3D.**

Źródło: Opracowanie własne – sprzęt dostępny projektantowi na czas przygotowania pracy inżynierskiej.

### **3.2 Opracowanie architektury systemu**

Następnie przechodzimy do rzeczy abstrakcyjnych i niewidocznych – do architektury samego systemu. Co klient powinien mieć możliwość zrobić podczas korzystania z projektowanego systemu? Biorąc pod uwagę, że duża część klientów to obcokrajowcy, jest potrzebna opcja wyboru języka na początku korzystania z aplikacji. Klient również powinien ustalić, czy będzie jadł na miejscu, czy bierze zamówienie na wynos. Dalej klient powinien mieć



możliwość wybrać spośród produktów pizzerii wyświetlonych na tablecie, czego dokładnie potrzebuje, zmodyfikować składniki wybranych dań, przejść do koszyka. W koszyku będzie miał możliwość na podsumowanie zamówienia, zmiany ilości produktów lub usunięcia jeżeli zdecyduje się na zakup czegoś innego. Następnie powinien dokonać płatności, po czym otrzymać paragon z numerem zamówienia i daniami, za które zapłacił.

Następnie zamówienie powinno być przetransportowane na kuchnię oraz do kasjera, natomiast ta część pracy będzie już zrealizowana w przyszłościowej wersji komercyjnej ze względu na brak dostępu do wewnętrznych systemów POS firmy [11].

Co stanie się, gdy w menu pojawi się nowe danie, które nie jest przewidziane przez programistę? Czy każdego razu trzeba będzie grzebać w kodzie i ręcznie dodawać każdy element? A co będzie gdy zamiast jednego stanowiska samoobsługowego będzie ich na placówce 5? Do każdego trzeba będzie się podłączać i odnawiać oprogramowanie? Nie jest to dopuszczalne w podobnych systemach.

Aby to obejść, dobrą praktyką jest przechowywanie informacji o wszystkich dostępnych daniach w chmurze z możliwością ich edycji. Aplikacja by wtedy przed każdym swoim włączeniem wysyłała zapytanie do chmury i pobierała ostatnią wersję menu, po czym opracowywała dane i generowała odpowiedź karteczki w odpowiednich kategoriach. Chmura powinna być dostępna w każdej chwili 24/7, więc należy trzymać bazę danych w miejscu ze stabilnym połączeniem do Internetu oraz prądu. Dobrym rozwiązaniem są serwery AWS Amazona (rysunek 10), korzystanie z których wybieramy dla realizacji projektu [12].



**Rysunek 10, Serwery AWS Amazon.**

Źródło: Strona internetowa [excelonsolutions.com](https://www.excelonsolutions.com/aws-services/): <https://www.excelonsolutions.com/aws-services/> , data aktualizacji: 10.05.2023.

Używając taką metodę generacji dań w połączeniu z wygodnym interfejsem użytkownika pojawia się możliwość dodania i edycji dań nie tylko wykwalifikowanemu informatykowi, ale również zwykłemu użytkownikowi/właścicielowi placówki, umiającemu obsługiwać komputer na podstawowym poziomie.

Podsumowując, system będzie się składał z głównego serwera z bazą danych wszystkich dań na placówce, przechowywanych w chmurze Amazonu. Właściciel placówki będzie miał do niej dostęp i będzie mógł edytować istniejące dania, dodawać nowe lub usuwać stare pozycje. Przed każdym włączeniem stanowiska do samoobsługi będzie wysyłane zapytanie na serwer, mające na celu pobrać najbardziej aktualną wersję menu rozpatrywanej restauracji. Niżej jest przedstawiony ogólny schemat działania systemu (rysunek 11).



**Rysunek 11, Schemat ideowy działania systemu.**

Źródło: Opracowanie własne stworzone na podstawie opisanej wyżej koncepcji systemu.

Cały projekt będzie napisany w środowisku programowania Visual Studio Code od Microsoftu [13].

## 4. Server KusTech

### 4.1 Wymagania

Wymagania, które powinien spełniać serwer w naszym projekcie są następujące:

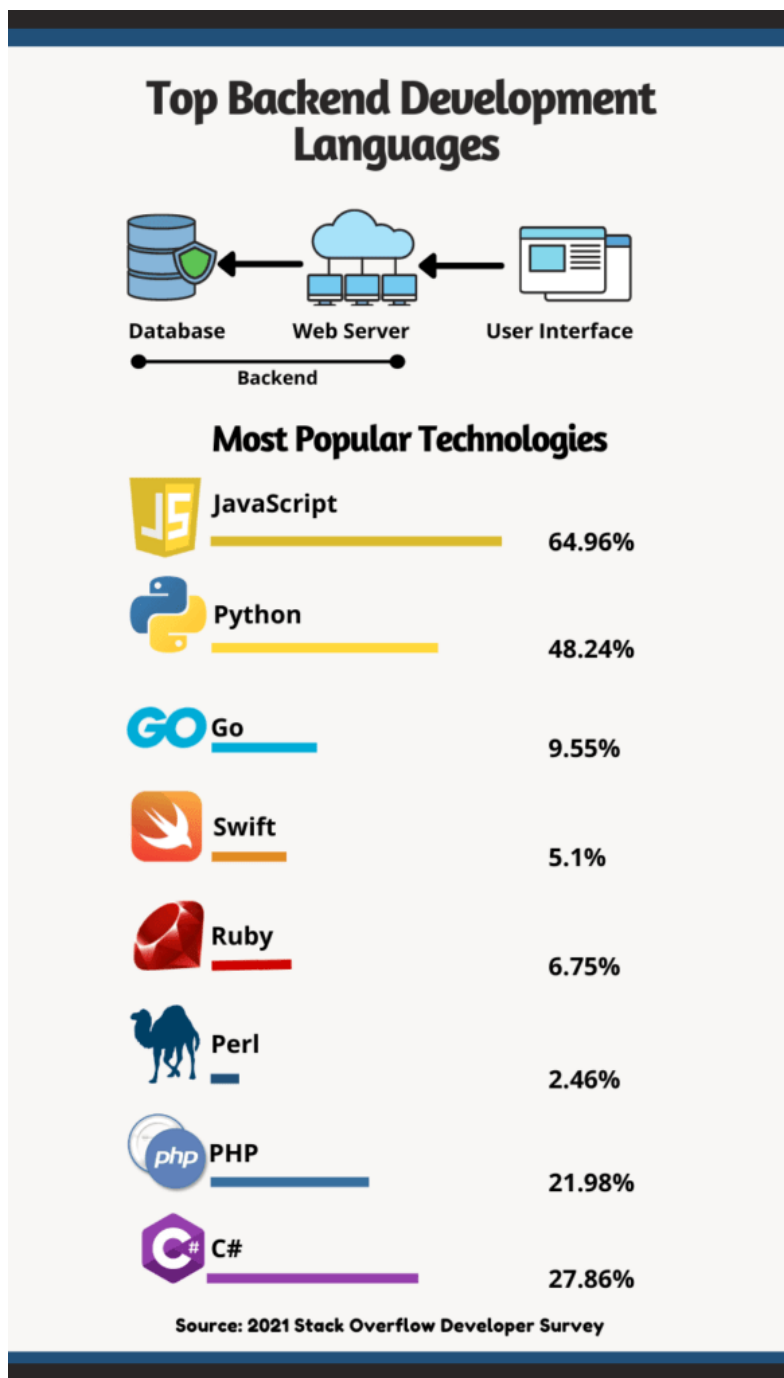
1. Powinien mieć bazę danych do przechowywania dań.
2. Powinien działać stabilnie i niezawodnie 24 godziny 7 dni w tygodniu.
3. Powinien mieć możliwość komunikacji ze światem zewnętrznym poprzez protokół API [14]
4. Powinien mieć interfejs użytkownika, pozwalający osobie niemającej wykształcenia informatycznego wprowadzać zmiany w bazie dań.
5. Powinien mieć sposób identyfikacji dla osób, upoważnionych do wejścia oraz redagowania zawartości bazy danych
6. Powinien mieć sposób autoryzacji dla zapytań API, wychodzących od stanowisk samoobsługowych

### 4.2 Technologia

Istnieje wiele języków programowania, pozwalających na napisanie aplikacji serwerowej. Najczęściej stosowanymi do tego językami są (rysunek 12):

1. JavaScript – jedno z najpopularniejszych narzędzi do tworzenia aplikacji serwerowych, zastosowane w dużej liczbie systemów informatycznych
2. Python – język o wysokiej produktywności, prosty w użyciu oraz posiadający wygodne narzędzia do tworzenia aplikacji webowych (m.in. Django, Flask itp.)
3. C# - język rozwijany przez Microsoft, który jest używany do tworzenia gier, aplikacji webowych oraz desktopowych.
4. PHP – jest to język popularny w napisaniu stron internetowych, posiadający potężną bazę programistów i frameworków.
5. Go – jest to język rozwijany i rozbudowywany przez Google, mający narzędzia do pisania stron internetowych
6. Ruby – język dynamiczny, interpretowany, obiektowy oraz mający zastosowanie w aplikacjach webowych
7. Swift – język programowania rozwijany przez Apple, stosowany w tworzeniu aplikacji serwerowych
8. Perl – interpretowany język programowania, którego używa się głównie do pisania skryptów oraz tworzenia narzędzi tekstowych. Jest on również często stosowany w tworzeniu aplikacji serwerowych, w szczególności do przetwarzania i analizowania dużych zbiorów danych oraz tworzenia narzędzi sieciowych.





Rysunek 12, Statystyka popularności stosowania języków programowania do napisania logiki serwerowej.

Źródło: Strona internetowa blog.back4app.com: <https://blog.back4app.com/backend-development-languages/>, data aktualizacji: 10.05.2023.

W związku z tym, że jedynym językiem programowania z powyżej wymienionej listy, z którym konstruktor miał pełny kontakt na studiach, jest Python, wybieram jego jako główny język aplikacji serwerowej [15].

Istnieją dwie główne biblioteki/frameworki dla języka Python, służące do projektowania aplikacji webowych – są to Django [16] oraz Flask [17]. Każda z nich ma swoje wady i zalety:

Flask jest zdecydowanie bardziej minimalistyczny od Django oraz jest łatwiejszy do nauki, dlatego jest odpowiednim wyborem dla projektów o niedużych wymaganiach oraz małej skali (rysunek 13). Flask pozwala projektantowi na zdecydowanie bardziej elastyczne podejście do napisania aplikacji webowych oraz zapewnia większą kontrolę nad jej wewnętrzną konfiguracją. Flask jest również lżejszy od Django (rysunek 14), co powoduje to, że aplikacje napisane we Flasku są szybsze oraz wydajniejsze.



**Rysunek 13, Biblioteka Flask Pythone.**

Źródło: Strona internetowa [pl.m.wikipedia.org](https://pl.m.wikipedia.org/wiki/Plik:Flask_logo.svg): [https://pl.m.wikipedia.org/wiki/Plik:Flask\\_logo.svg](https://pl.m.wikipedia.org/wiki/Plik:Flask_logo.svg) , data aktualizacji: 10.05.2023.

Z drugiej strony Django jest pełnoprawnym frameworkiem, który ma do zaoferowania pełen zestaw narzędzi i funkcjonalności do napisania skomplikowanych aplikacji webowych. Django posiada w sobie wiele wbudowanych funkcji standardowych, które pozwalają na szybkie i łatwe tworzenie zaawansowanych aplikacji serwerowych. Framework ten charakteryzuje się mnóstwem gotowych rozwiązań, takich jak autoryzacja użytkowników, obsługa formularzy oraz obsługa bazy danych.

# django

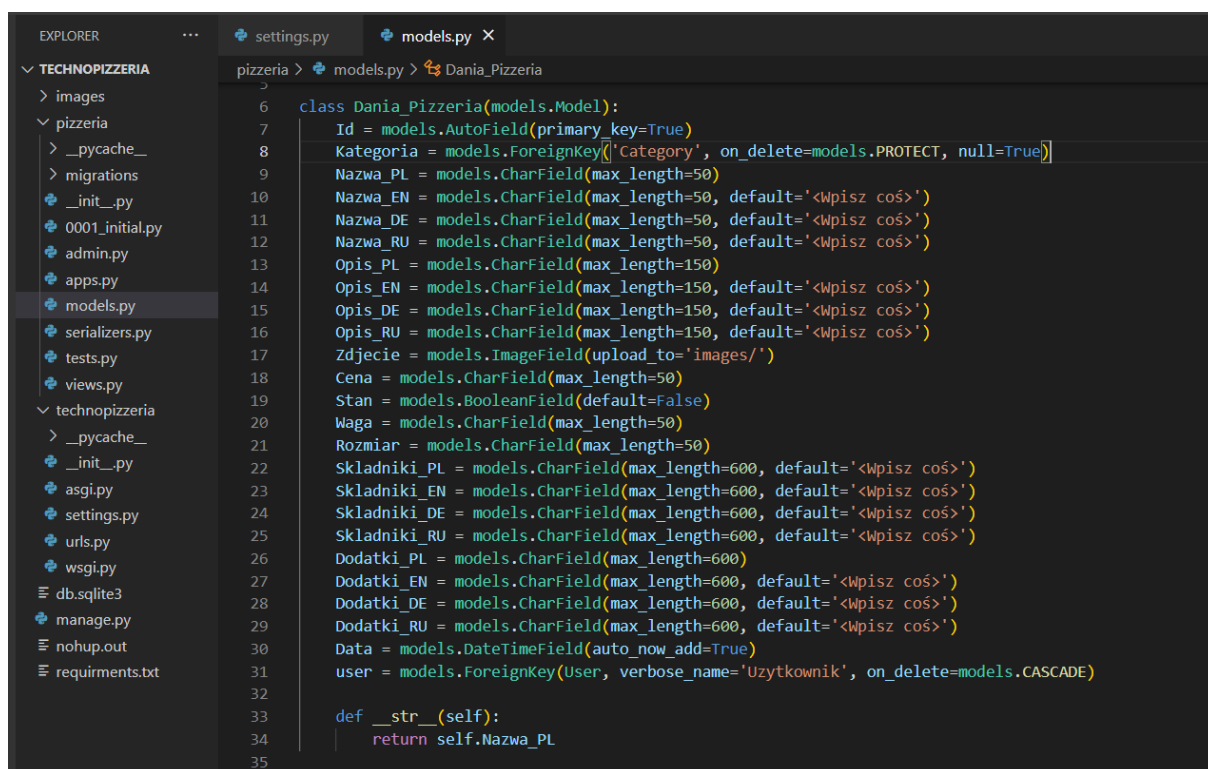
**Rysunek 14, Biblioteka Django Pythone.**

Źródło: Strona internetowa [www.djangoproject.com](https://www.djangoproject.com/community/logos/): <https://www.djangoproject.com/community/logos/> , data aktualizacji: 10.05.2023.

W związku z tym, że ostatnio rozpatrywany framework posiada wiele gotowych modułów, zwłaszcza do pracy z bazą danych oraz autoryzacją użytkowników, ostateczny wybór to Django. Brak potrzeby napisania modułów od zera również zwiększa prędkość napisania aplikacji, zwiększa jakość kodu oraz pozwala projektantowi skupić się na logice działania aplikacji.

### 4.3 Proces projektowy

Cały system będzie oparty na bazie danych, więc należy ją zaprojektować we właściwy sposób. W projektach Django dla projektowania konfiguracji baz danych używane jest plik mający nazwę „models.py” (załącznik 1). W nim należy dodać „class” z nazwą naszej bazy oraz wypełnić go polami potrzebnymi do zapisania wszystkich niezbędnych informacji o produkcie. W naszym przypadku są to pola tekstowe, odwołania do innych baz, pola zdjęciowe, liczbowe oraz pole z datą dodania (rysunek 15). Podczas projektowania zostały uwzględnione 4 wersje językowe – polska, angielska, niemiecka oraz rosyjska.



```
6 class Dania_Pizzeria(models.Model):
7     Id = models.AutoField(primary_key=True)
8     Kategoria = models.ForeignKey('category', on_delete=models.PROTECT, null=True)
9     Nazwa_PL = models.CharField(max_length=50)
10    Nazwa_EN = models.CharField(max_length=50, default='<wpisz coś>')
11    Nazwa_DE = models.CharField(max_length=50, default='<wpisz coś>')
12    Nazwa_RU = models.CharField(max_length=50, default='<wpisz coś>')
13    Opis_PL = models.CharField(max_length=150)
14    Opis_EN = models.CharField(max_length=150, default='<wpisz coś>')
15    Opis_DE = models.CharField(max_length=150, default='<wpisz coś>')
16    Opis_RU = models.CharField(max_length=150, default='<wpisz coś>')
17    Zdjecie = models.ImageField(upload_to='images/')
18    Cena = models.CharField(max_length=50)
19    Stan = models.BooleanField(default=False)
20    Waga = models.CharField(max_length=50)
21    Rozmiar = models.CharField(max_length=50)
22    Skladniki_PL = models.CharField(max_length=600, default='<wpisz coś>')
23    Skladniki_EN = models.CharField(max_length=600, default='<wpisz coś>')
24    Skladniki_DE = models.CharField(max_length=600, default='<wpisz coś>')
25    Skladniki_RU = models.CharField(max_length=600, default='<wpisz coś>')
26    Dodatki_PL = models.CharField(max_length=600)
27    Dodatki_EN = models.CharField(max_length=600, default='<wpisz coś>')
28    Dodatki_DE = models.CharField(max_length=600, default='<wpisz coś>')
29    Dodatki_RU = models.CharField(max_length=600, default='<wpisz coś>')
30    Data = models.DateTimeField(auto_now_add=True)
31    user = models.ForeignKey(User, verbose_name='Uzytkownik', on_delete=models.CASCADE)
32
33    def __str__(self):
34        return self.Nazwa_PL
35
```

Rysunek 15, Struktura wewnętrzna obiektu w modelu bazy danych z daniami.

Źródło: Opracowanie własne.

Każdy obiekt w bazie danych dań zawiera 25 pól:

1. Id – Indywidualny identyfikator obiektu, wydawany podczas jego utworzenia.
2. Kategoria – Kategoria, do której będzie należało utworzone danie. Kategorie są tylko cztery – Pizza, Napoje, Sałatki i Nowości, więc aby każdego razu ręcznie nie

wpisywać nazwy, został utworzony jeszcze jeden model, do którego jest w tej bazie odwołanie.

3. Nazwa\_PL – Nazwa produktu po polsku.
4. Nazwa\_EN – Nazwa produktu po angielsku.
5. Nazwa\_DE – Nazwa produktu po niemiecku.
6. Nazwa\_RU – Nazwa produktu po rosyjsku.
7. Opis\_PL – Opis produktu po polsku.
8. Opis\_EN – Opis produktu po angielsku.
9. Opis\_DE – Opis produktu po niemiecku.
10. Opis\_RU – Opis produktu po rosyjsku.
11. Zdjęcie – Miejsce dla zdjęcia, które będzie się wyświetlało obok dania w menu.
12. Cena – Cena w złotych.
13. Stan – Czy mamy aktualnie ten produkt w menu? Zamiast usunięcia i dodania produktu sezonowego można w tym polu odznaczyć, że jego nie będzie przez jakiś czas.
14. Waga – Waga produktu w gramach.
15. Rozmiar – Rozmiar produktu. W naszym przypadku rozmiar pizzy.
16. Składniki\_PL – Składniki, które mogą w tym daniu występować po polsku.
17. Składniki\_EN – Składniki, które mogą w tym daniu występować po angielsku.
18. Składniki\_DE – Składniki, które mogą w tym daniu występować po niemiecku.
19. Składniki\_RU – Składniki, które mogą w tym daniu występować po rosyjsku.
20. Dodatki\_PL – Możliwe dodatki do dania po polsku.
21. Dodatki\_EN – Możliwe dodatki do dania po angielsku.
22. Dodatki\_DE – Możliwe dodatki do dania po niemiecku.
23. Dodatki\_RU – Możliwe dodatki do dania po rosyjsku.
24. Data – Data dodania zamówienia przez kierownika placówki – informacja statystyczna
25. User – Informacja o użytkowniku, który tą pozycję dodał – informacja statystyczna

Poza aktualizacją menu należy też przemyśleć aktualizację wyglądu aplikacji – na wypadek gdy pojawią się nowe promocje oraz przy zmianie wewnętrznego wyglądu placówki. Aby ten moment uwzględnić, został dodany jeszcze jeden model z obrazkami statycznymi, które również się pobierają przed każdym włączeniem aplikacji (rysunek 16).



**Rysunek 16, Struktura wewnętrzna obiektu modelu z konfiguracją początkową.**

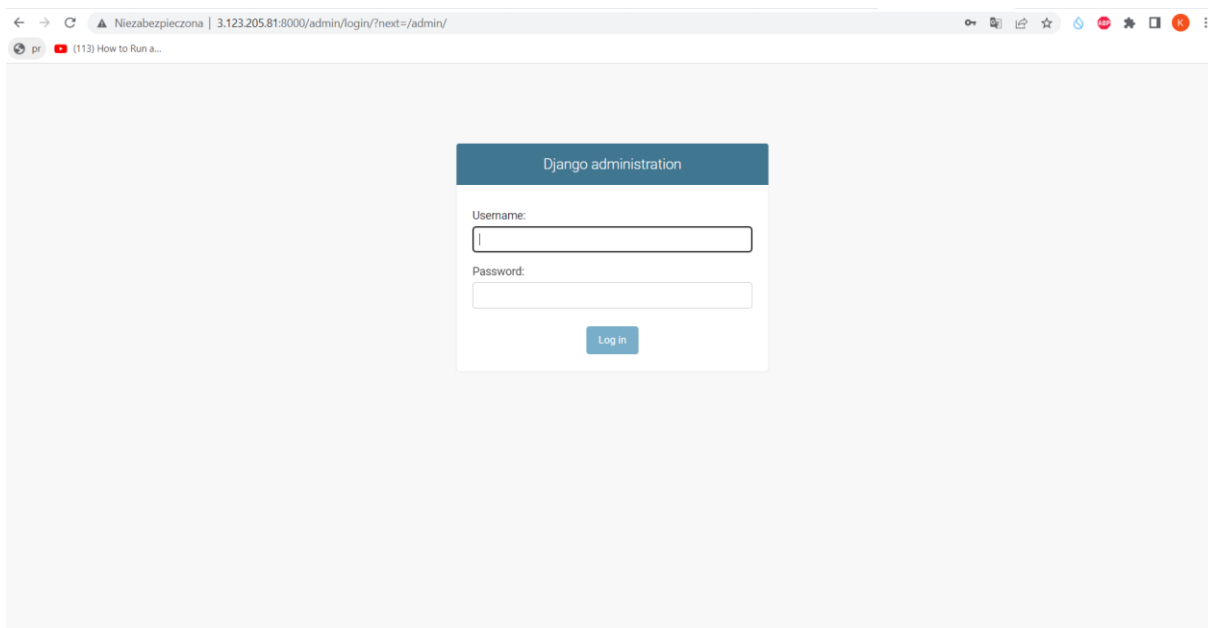
Źródło: Opracowanie własne.

W tym modelu jest tylko jeden obiekt, który zawiera 8 pól:

1. name – Nazwa pomocnicza, jest potrzebna tylko dla wewnętrznych algorytmów systemu.
2. first\_image – Zdjęcie wstępne, witające użytkownika w aplikacji.
3. second\_image – Zdjęcie na następnej wkładce aplikacji.
4. icon\_na\_miejscu – Zdjęcie znajdujące się na przycisku wyboru miejsca zjedzenia dania. (Na miejscu).
5. icon\_na\_wynos – Zdjęcie znajdujące się na przycisku wyboru miejsca zjedzenia dania. (Na wynos).
6. icon\_dodatki – Zdjęcie znajdujące się na przycisku z możliwymi dodatkami do dania.
7. icon\_karta – Zdjęcie znajdujące się na przycisku płatności kartą.
8. icon\_blik – Zdjęcie znajdujące się na przycisku płatności Blikiem.

Mając te modele, należy zorganizować do nich dostęp z poziomu właściciela pizzerii.

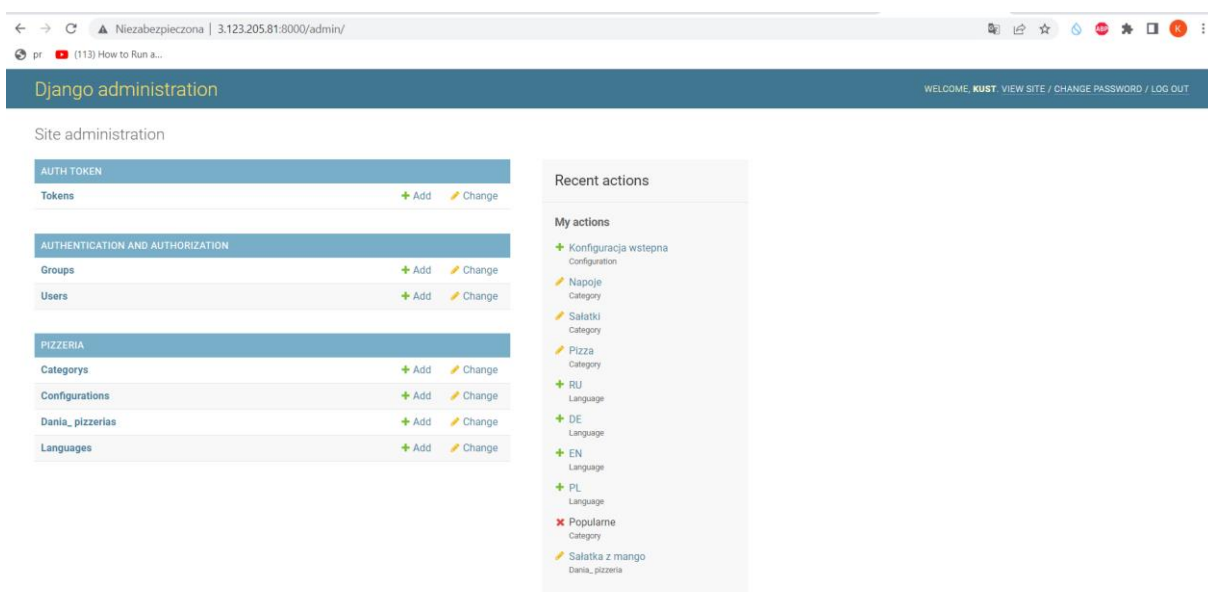
Również należy zadbać o system autoryzacji dla tej osoby, aby zapobiec niekontrolowanej edycji dań przez osoby nieupoważnione. Podłączamy panel administracyjny oraz dodajemy do niego blok logowania Django (rysunek 17):



**Rysunek 17, Logowanie do panelu administracyjnego na Django.**

Źródło: Opracowanie własne znajduje się pod adresem <http://3.123.205.81:8000/admin/> , data aktualizacji: 10.05.2023

Po wprowadzeniu danych logowania właściciela pizzerii otrzymujemy dostęp do panelu administracyjnego [18] (rysunek 18):

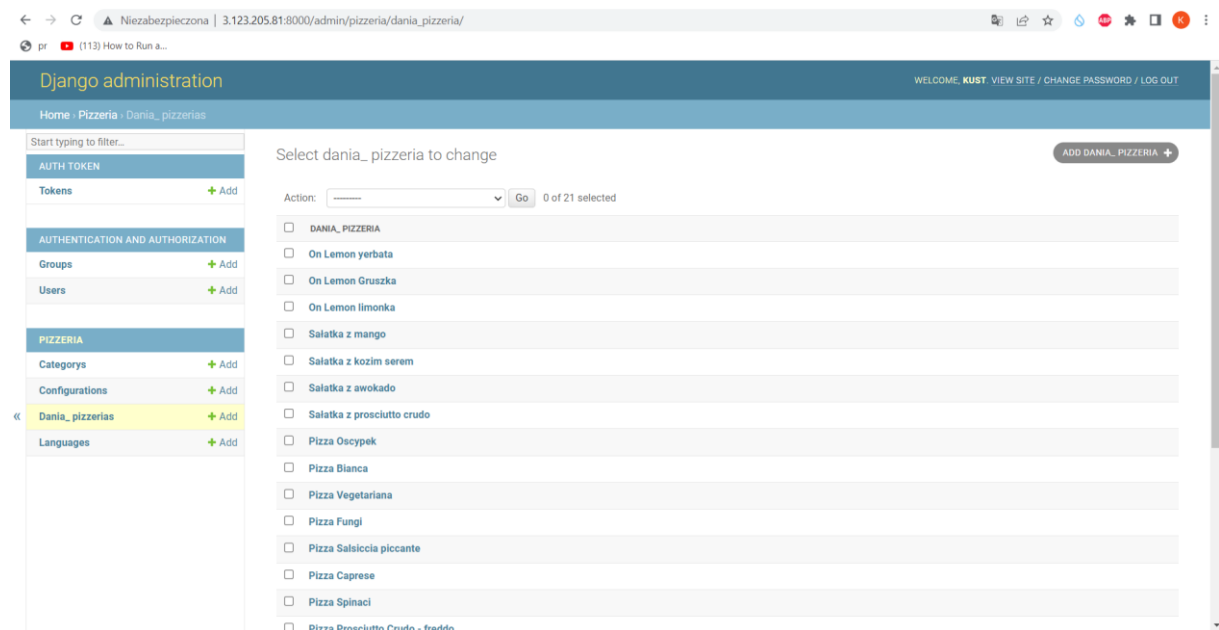


**Rysunek 18, Panel administracyjny na Django.**

Źródło: Opracowanie własne znajduje się pod adresem <http://3.123.205.81:8000/admin/> , data aktualizacji: 10.05.2023.

Z poziomu panelu administracyjnego otrzymujemy dostęp do wyżej wymienionych modeli, modelu „Users” z użytkownikami [19] oraz do modelu „Tokens” z tokenami [20] dostępu do API naszej bazy. W modelu „Users” znajdują się wszyscy użytkownicy mający dostęp do serwera z ich danymi logowania. W modelu „Tokens” znajdują się wszystkie kody

autoryzacyjne logowania API, które aplikacje zewnętrzne muszą wysyłać wraz z próbą pobierania danych o daniach z naszej bazy. Przedstawienie graficzne zawartości bazy wygląda w sposób następujący (rysunek 19):



**Rysunek 19, Zawartość modelu bazy danych z daniami.**

Źródło: Opracowanie własne znajduje się pod adresem

[http://3.123.205.81:8000/admin/pizzeria/dania\\_pizzeria/](http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/) , data aktualizacji: 10.05.2023.

Są tutaj przedstawione wszystkie pozycje w menu, które będą demonstrowane użytkownikowi podczas wyboru dań na stanowisku samoobsługowym. Z tego poziomu system pozwala na usunięcie lub dodanie nowych pozycji. Każdy obiekt w środku ma kilka pól, które użytkownik może edytować (rysunek 20):

Home Pizzeria Dania\_pizzerias Pizza Margherita

Start typing to filter...

AUTH TOKEN

Tokens [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

PIZZERIA

Categories [+ Add](#)

Configurations [+ Add](#)

Dania\_pizzerias [+ Add](#)

Languages [+ Add](#)

Change dania\_pizzeria

Pizza Margherita

Kategoria: Pizza

Nazwa PL: Pizza Margherita

Nazwa EN: <Wpisz coś>

Nazwa DE: <Wpisz coś>

Nazwa RU: <Wpisz coś>

Opis PL: z sosem pomidorowym i mozzarellą di latte

Opis EN: <Wpisz coś>

Opis DE: <Wpisz coś>

Opis RU: <Wpisz coś>

Zdjecie: Currently: images/Pizza\_Margherita\_Qey4zHT.png  
Change: [Wybierz plik](#) Nie wybrano pliku

Cena: 29/34/39

**Rysunek 20, Zawartość obiektu z modelu bazy danych z daniami (1).**

Źródło: Opracowanie własne znajduje się pod adresem

[http://3.123.205.81:8000/admin/pizzeria/dania\\_pizzeria/3/change/](http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/3/change/) , data aktualizacji: 10.05.2023.

Po edycji pozycji należy nacisnąć przycisk „SAVE” w prawym dolnym rogu, po czym zmiany zostaną zapisane w modelu (rysunek 21).

Start typing to filter...

AUTH TOKEN

Tokens [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

PIZZERIA

Categories [+ Add](#)

Configurations [+ Add](#)

Dania\_pizzerias [+ Add](#)

Languages [+ Add](#)

Change dania\_pizzeria

Pizza Margherita

☒ Stan

Waga: 200

Rozmiar: 29/34/39

Skladniki PL: sos pomidorowy/mozzarella fior di latte

Skladniki EN: <Wpisz coś>

Skladniki DE: <Wpisz coś>

Skladniki RU: <Wpisz coś>

Dodatki PL: cebula (+5,00 zł), cukinia (+5,00 zł), oliwki (+5)

Dodatki EN: <Wpisz coś>

Dodatki DE: <Wpisz coś>

Dodatki RU: <Wpisz coś>

Uzytkownik: kust

Delete

Save and add another

Save and continue editing

SAVE

**Rysunek 21, Zawartość obiektu z modelu bazy danych z daniami (2).**

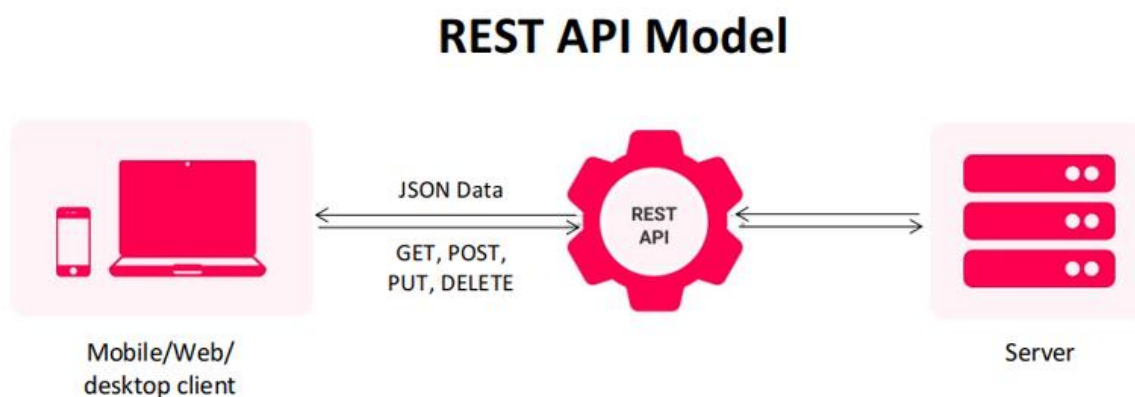
Źródło: Opracowanie własne znajduje się pod adresem

[http://3.123.205.81:8000/admin/pizzeria/dania\\_pizzeria/3/change/](http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/3/change/) , data aktualizacji: 10.05.2023.

Po utworzeniu bazy danych należy umożliwić komunikację z nią zewnętrzną aplikacją – w moim przypadku stanowisku samoobsługowemu. Biblioteka Django bez dodatkowych modułów nie pozwala na komunikację przez API, dlatego potrzebuje dodatkowej biblioteki. Na rolę odpowiedniej biblioteki może pretendować Rest API [21], która nam umożliwi łatwe



tworzenie interfejsów API, pozwalających na komunikację z naszą bazą danych (rysunek 22).



**Rysunek 22, Schemat działania protokołu API.**

Źródło: Strona internetowa mindinventory.com: <https://www.mindinventory.com/blog/best-practices-rest-api-development/> , data aktualizacji: 10.05.2023.

Rest API jest popularnym protokołem komunikacyjnym, który funkcjonuje na zasadzie zapytanie-odpowiedź. Używając Rest API będziemy w stanie wykonywać różne operacje na bazie danych, takie jak odczyt (GET), zapis (POST), aktualizacja (PUT) i usuwanie (DELETE) rekordów. Po wysłaniu zapytania dane będą zwracane w formacie JSON, który jest popularnym standardem wymiany danych pomiędzy aplikacjami.

Poza samym użyciem Rest API należy jeszcze dodać autoryzację wchodzących zapytań do bazy. Do tego będę używał modułu biblioteki Django REST framework pod nazwą „rest\_framework.auth\_token”, który umożliwia autoryzację za pomocą tokenów autoryzacyjnych. Ta metoda jest często stosowana wtedy, gdy aplikacja kliencka (w moim przypadku stanowisko samoobsługowe) musi uzyskać dostęp do zasobów chronionych przez autentyfikację (rysunek 23) (załącznik 2).

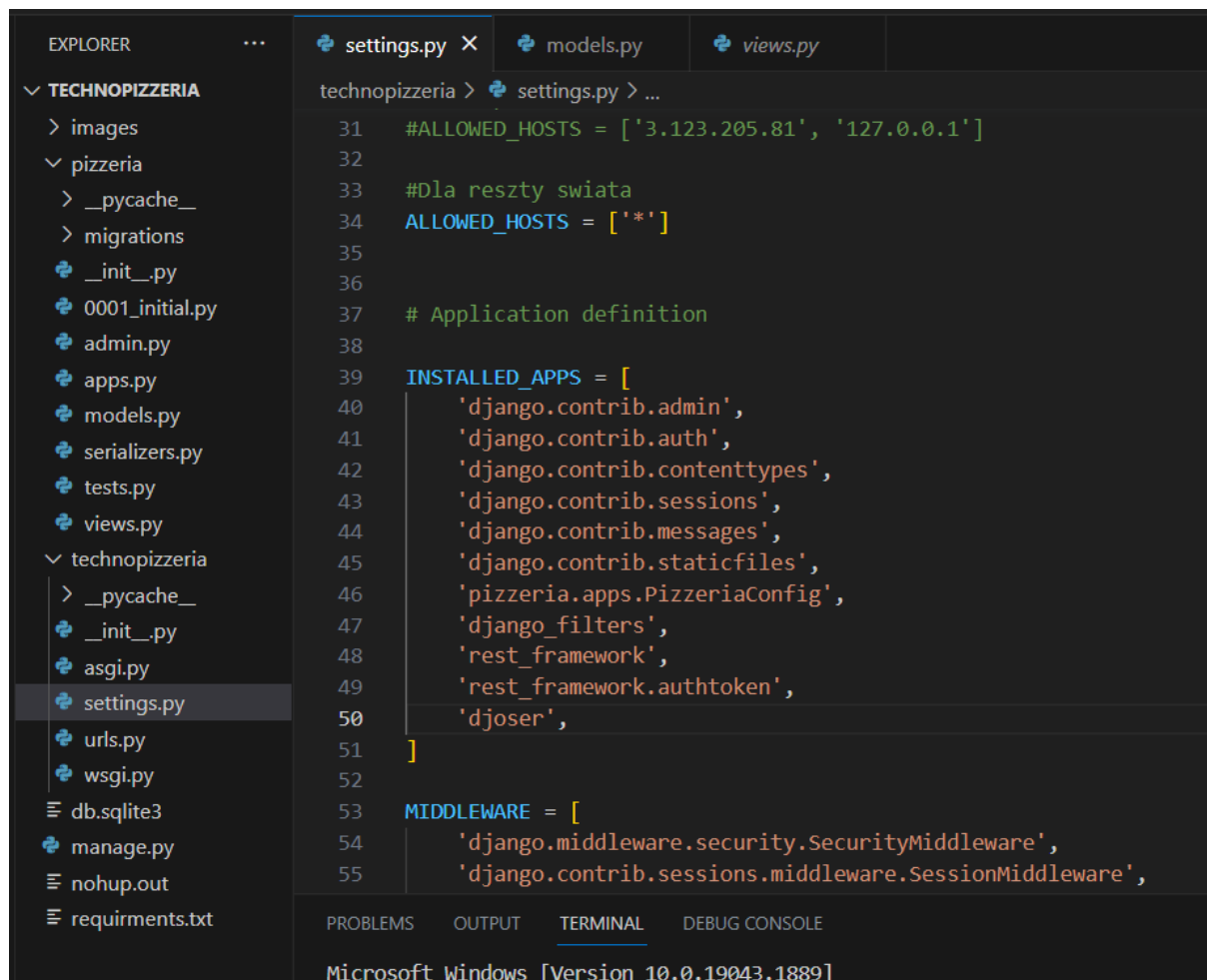
```
3 def get_meals(Token):
4     import requests
5     import json
6
7     token = 'Token ' + str(Token)
8
9     #WEB
10    r = requests.get('http://3.123.205.81:8000/api/v1/pizza/', headers={'Authorization': token})
11
12    #LOCAL
13    #r = requests.get('http://127.0.0.1:8000/api/v1/pizza/', headers={'Authorization': token})
14
15    r_dict = json.loads(r.text)
```

**Rysunek 23, Wysyłanie zapytania na serwer z poziomu aplikacji.**

Źródło: Opracowanie własne.

Działanie autoryzacji korzystając z tokenów, polega na utworzeniu i udostępnieniu użytkownikowi unikalnego tokena, który jest zapisany w bazie danych (Model „Tokens”). Klient powinien wysłać otrzymany token wraz z każdym wysłaniem zapytania, aby uzyskać dostęp do zawartości bazy.

Wszystkie wymienione moduły oraz moduły Django, wymagane do poprawnego działania opisanego wyżej systemu znajdują się w pliku „settings.py” projektu Django (rysunek 24):



```
technopizzeria > settings.py > ...
31  #ALLOWED_HOSTS = ['3.123.205.81', '127.0.0.1']
32
33  #Dla reszty swiata
34  ALLOWED_HOSTS = ['*']
35
36
37  # Application definition
38
39  INSTALLED_APPS = [
40      'django.contrib.admin',
41      'django.contrib.auth',
42      'django.contrib.contenttypes',
43      'django.contrib.sessions',
44      'django.contrib.messages',
45      'django.contrib.staticfiles',
46      'pizzeria.apps.PizzeriaConfig',
47      'django_filters',
48      'rest_framework',
49      'rest_framework.authtoken',
50      'djoser',
51  ]
52
53  MIDDLEWARE = [
54      'django.middleware.security.SecurityMiddleware',
55      'django.contrib.sessions.middleware.SessionMiddleware',
```

**Rysunek 24, Moduły potrzebne do działania aplikacji serwerowej.**  
Źródło: Opracowanie własne.

## 4.4 Serwer

Mając zaprojektowaną stronę należy zadbać o jej stabilne działanie oraz dostęp do niej na żądanie powstającego stanowiska samoobsługowego. Dla tego celu należy znaleźć serwer, na którym ją розміścimy.

Serwer może działać lokalnie lub być podłączonym do Internetu - każda z tych metod ma swoje wady i zalety. Jeżeli strona będzie umieszczona na lokalnym serwerze w sieci wewnętrznej, to zdecydowanie większe będzie bezpieczeństwo ze względu na brak dostępu do niej osób spoza zasięgu sygnału WIFI w naszej placówce. Rozmieszczenie strony na serwerze z podłączeniem do Internetu pozwoli na dostęp do niej spoza granicy placówki, co zmniejsza bezpieczeństwo, natomiast zwiększa wygodę jej obsługi (na przykład można

zalogować się z domu) oraz pozwala na większą skalowalność systemu. Na potrzeby naszego projektu wybrałem rozmieszczenie na serwerze z dostępem do Internetu.

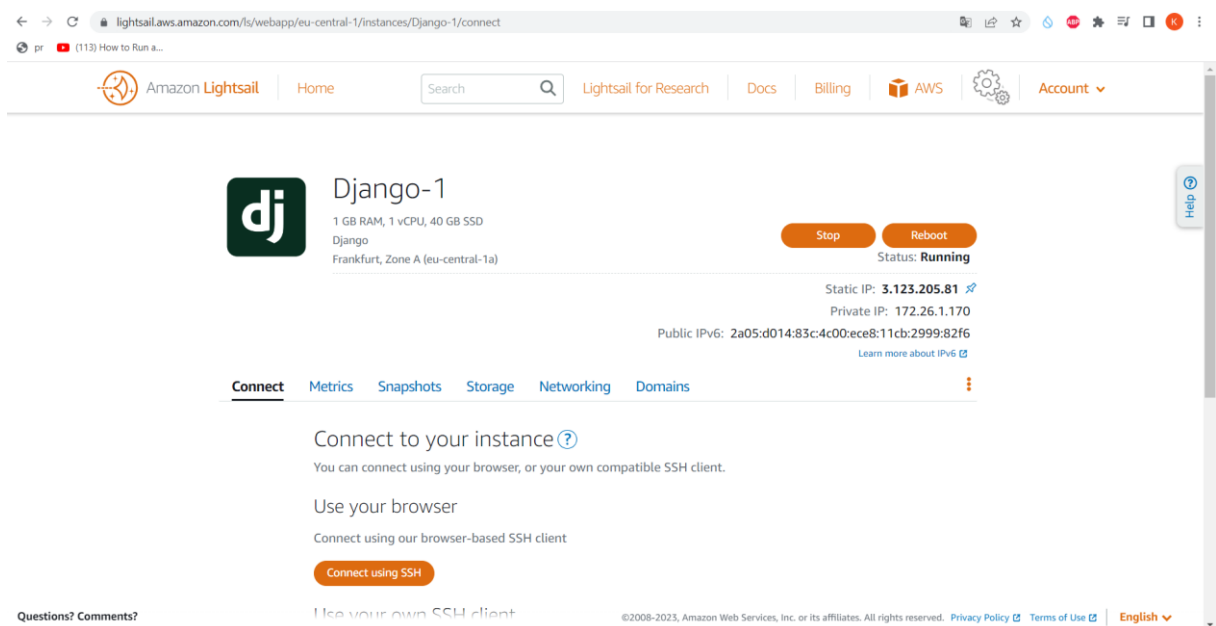


**Rysunek 25, Szafy serwerowe.**

Źródło: Strona internetowa bestmicro.pl: [https://bestmicro.pl/blog/15\\_serwery-co-warto-o-nich-wiedziec.html](https://bestmicro.pl/blog/15_serwery-co-warto-o-nich-wiedziec.html) , data aktualizacji: 10.05.2023.

Kolejnym krokiem należy zdecydować czy strona będzie rozmieszczona na jakimś fizycznym urządzeniu (komputer, laptop, itp.) czy będę korzystał z rozwiązań chmurowych dużych korporacji (AWS, Azure itp.) (rysunek 25). Korzystanie z serwerów w chmurze ma zdecydowanie większy potencjał ze względu na ich specjalnie wybudowaną infrastrukturę dla serwerów oraz małe koszty utrzymania wirtualnej maszyny.

Wybrałem chmurę AWS Amazonu ze względu na cenę - abonament na moce obliczeniowe pasujące dla powstającego systemu kosztują tylko 5\$ miesięcznie [22] (rysunek 26).



**Rysunek 26, Wykupiony serwer AWS do funkcjonowania aplikacji webowej.**

Źródło: Opracowanie własne bazujące na stronie [lightsail.aws.amazon.com](https://lightsail.aws.amazon.com):

<https://lightsail.aws.amazon.com/ls/webapp/eu-central-1/instances/Django-1/connect> , *data aktualizacji:*  
10.05.2023.

Na potrzebę projektu został wykupiony serwer na systemie operacyjnym Ubuntu[23], na którym została rozmieszczona zaprojektowana przeze mnie strona. Podłączenie jest przez SSH[24], co jest bardzo wygodnym sposobem komunikacji z serwerem zdalnym. Dodatkowa domena nie została wykupiona ze względu na to, że głównie strona będzie używana przez protokoły API na rzecz dostępu do bazy danych. W wersji komercyjnej należy wykupić domenę.

## 5. Aplikacja KusTech

### 5.1 Wymagania

Wymagania, które powinna spełniać aplikacja w powstającym projekcie, są następujące:

1. Powinna być kompatybilna z systemem operacyjnym Android.
2. Powinna działać stabilnie bez potrzeby resetowania co jakiś czas.
3. Powinna mieć przejrzysty i intuicyjny interfejs użytkownika.
4. Powinna mieć możliwość integracji z systemami płatniczymi.
5. Powinna mieć możliwość personalizacji zamówień.
6. Powinna mieć możliwość integracji z innymi systemami.

### 5.2 Technologia

Istnieje wiele języków, które pozwalają na napisanie aplikacji na Androidzie. Większość z nich już była omówiona przy doborze języka do napisania aplikacji serwerowej – tam podjęto decyzję o stosowaniu Pythonie Django. Do napisania aplikacji użyję kolejnego frameworku Pythona pod nazwą Kivy [25] oraz jego rozszerzenia w stylu Material Design – KivyMD [26].

Kivy (rysunek 27) jest otwartą biblioteką do tworzenia aplikacji, bazującą na języku Python. Jest to wieloplatformowy framework, który pozwala na napisanie aplikacji mobilnych, desktopowych oraz innych interaktywnych aplikacji z GUI (graficznym interfejsem użytkownika). Kivy oferuje obszerną ilość funkcji, takich jak obsługa gestów dotykowych, animacje, grafika 2D i 3D, obsługa wielu okien, dźwięk oraz wiele innych. Kivy funkcjonuje na różnych platformach, w tym na systemach Windows, Mac OS X, Linux, Android i iOS.



**Rysunek 27, Biblioteka Kivy Python.**

Źródło: Strona internetowa lowendplay.com: <https://lowendplay.com/python-kivy/>, data aktualizacji: 10.05.2023.

KivyMD (Kivy Material Design) (rysunek 28) to rozszerzenie dla Kivy, które pozwala na użycie wzornictwa Material Design stworzonego przez korporację Google. KivyMD posiada w sobie gotowe do użycia elementy interfejsu użytkownika, takie jak przyciski, pola tekstowe,

okna dialogowe, ikony, menu i wiele innych, które są zgodne ze postulatami Material Designu. KivyMD służy do projektowania aplikacji zgodnych z najnowszymi standardami wzornictwa, co pozwala na znacznie szybsze i łatwiejsze tworzenie aplikacji mobilnych, desktopowych i innych.



**Rysunek 28, Biblioteka KivyMD Python.**

Źródło: Strona internetowa github.com : <https://repository-images.githubusercontent.com/284716598/5af8e880-d8ba-11ea-9ce1-e5e8d603143f>, data aktualizacji: 10.05.2023.

Aplikacja będzie projektowana na systemie Windows, po czym poprzez specjalne narzędzie zostanie przekonwertowana do pliku Android .apk.

### 5.3 Proces projektowy

Architektura Kivy składa się z 2 plików – pliku Pythona z logiką działania aplikacji oraz z plikiem .kv [27] służącym do projektowania graficznego interfejsu użytkownika.

Aplikacja będzie się składała z 7 ekranów:

1. Ekran\_witający – Ekran witający użytkownika.
2. Ekran\_wyboru\_miejsca – Ekran służący do wyboru miejsca, gdzie klient będzie jadł.
3. Ekran\_menu – Główny ekran służący do wyboru dań. Zawiera 4 stałe kategorie oraz miejsce do wygenerowanych dań.
4. Ekran\_koszyka – Ekran koszyka, zawierający wybrane wcześniej dania.
5. Ekran\_wyboru\_płatności – Ekran służący do wyboru płatności pomiędzy kartą a blikiem
6. Ekran\_płatności – Ekran z przyciskiem, po naciśnięciu którego jest symulowana właściwie przeprowadzona płatność
7. Ekran\_końcowy – Ekran służący do wyświetlenia numeru zamówienia oraz świadczący, że płatność się udała.



Pierwsze co robi aplikacja przed włączeniem – wysyła zapytanie API wraz z tokenem autoryzacyjnym do wyżej opisanego serwera z bazą danych ustawień początkowych, po czym wysyła zapytanie API na serwer z bazą danych dań, aby pobrać najnowsze informacje, dotyczące produktów z menu. Po ich otrzymaniu w formacie JSON [28] przekształca ich do formatu słownika (dict). Po czym już rozpoczyna się generacja interfejsu użytkownika:

### Ekran\_witający

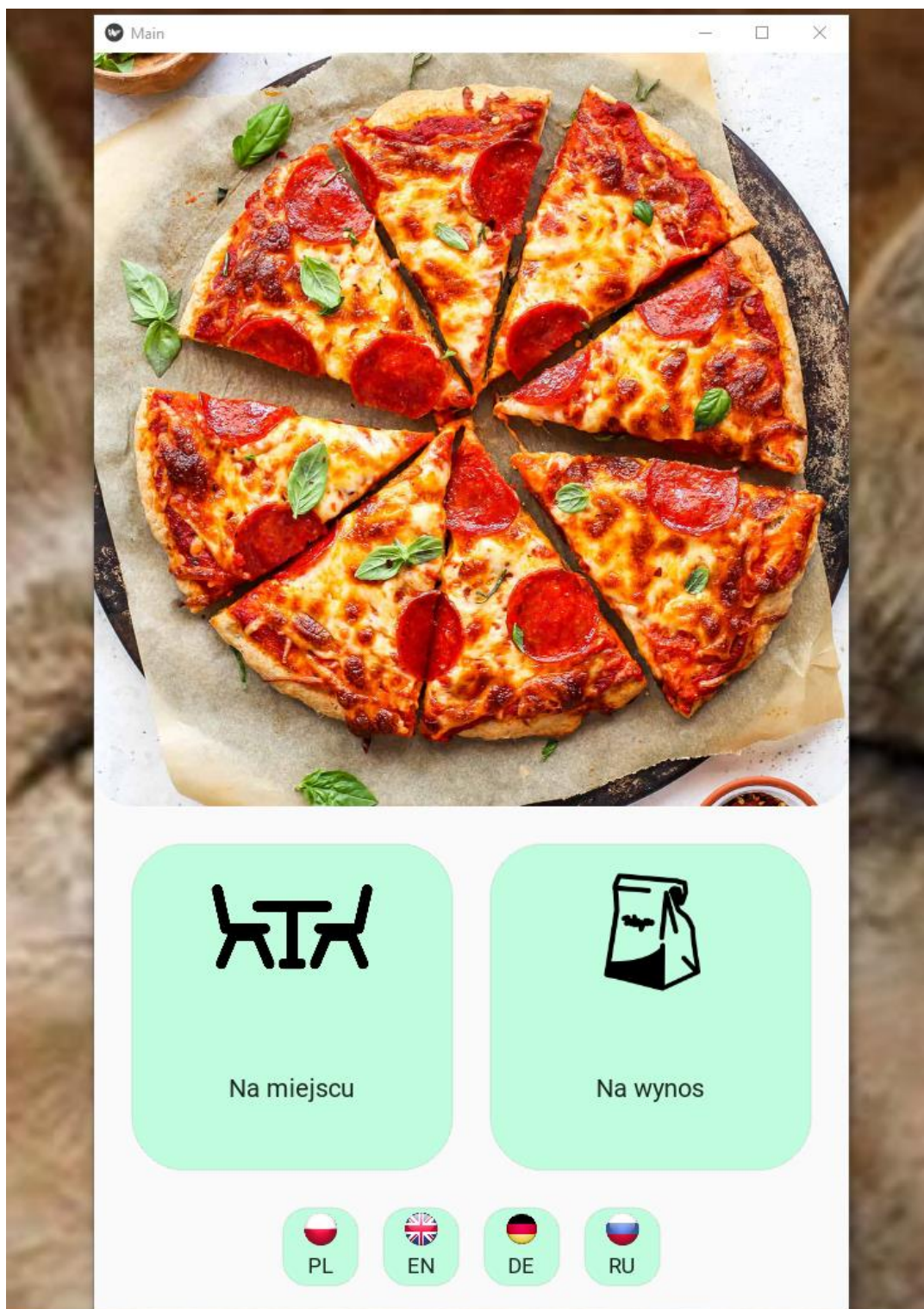


**Rysunek 29, Ekran witający użytkownika.**

Źródło: Opracowanie własne.

Jest to pierwszy ekran (rysunek 29), z którym klient ma do czynienia gdy podchodzi do mojego stanowiska samoobsługi. Może wybrać język komunikacji z aplikacją – również w tym języku będą wyświetlane wszystkie pozycje w menu.

## Ekran\_wyboru\_miejsca

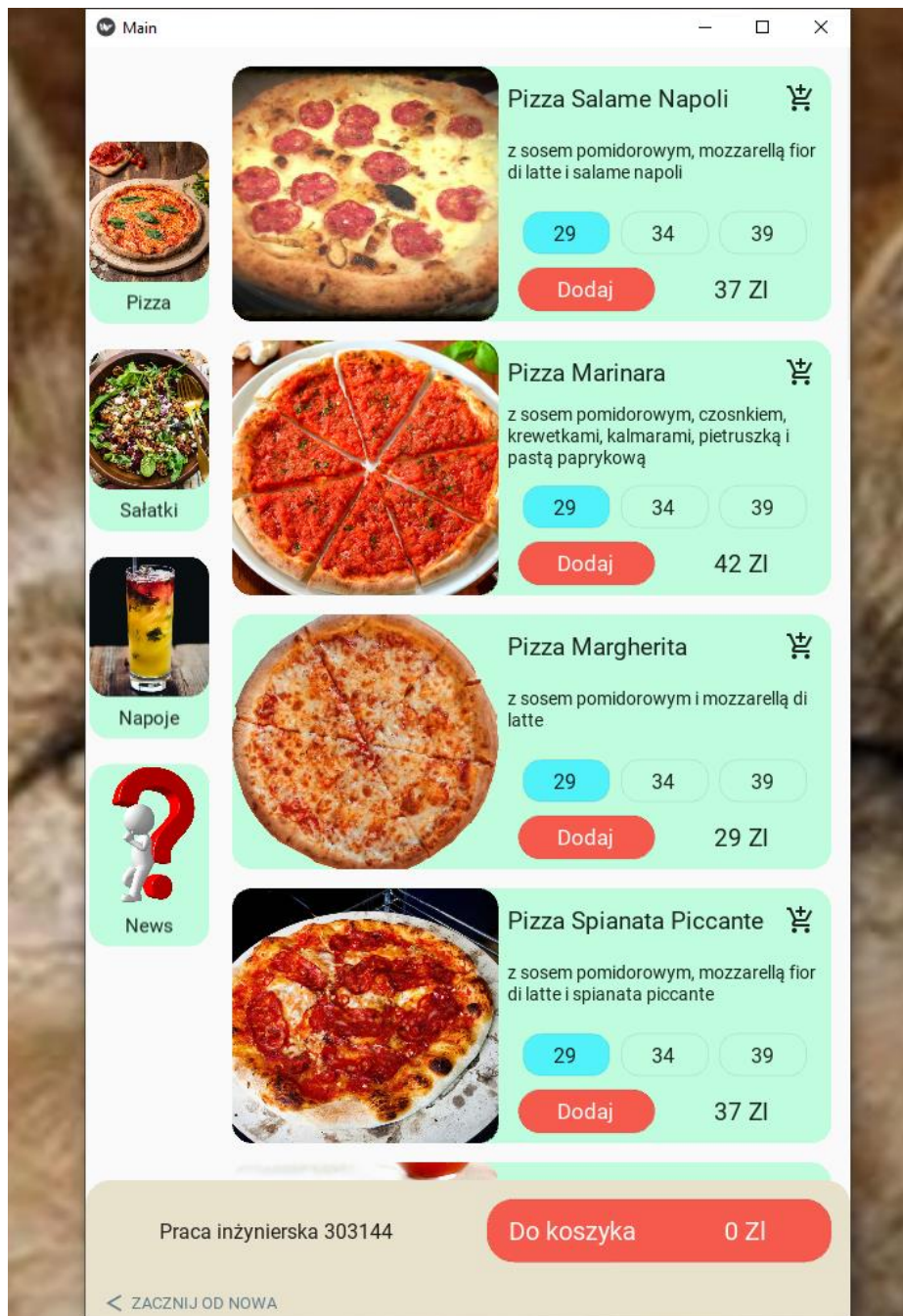


**Rysunek 30, Ekran wyboru miejsca.**  
Źródło: Opracowanie własne.

Drugi ekran (rysunek 30) jest wprowadzony po to, żeby użytkownik miał możliwość wyboru miejsca, gdzie zadecyduje zjeść zamówienie. Również jest tu przewidziana opcja wyboru języka na wypadek, gdy klient nie zdąży tego zrobić przy pierwszym podejściu.



## Ekran\_menu

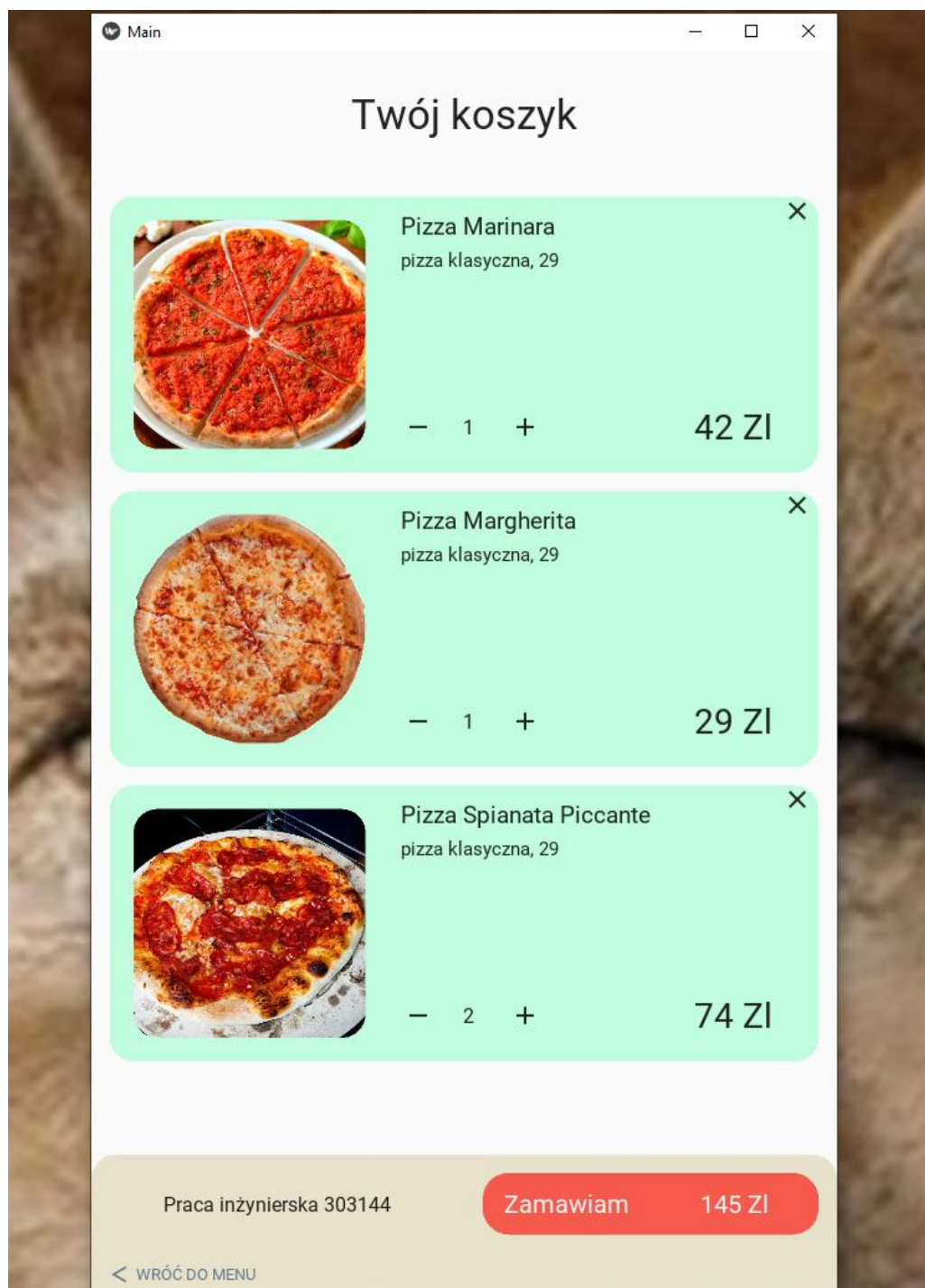


**Rysunek 31, Ekran menu.**  
Źródło: Opracowanie własne.

Na ekranie z menu (rysunek 31) znajdują się 4 stałe kategorie dań „Pizza”, „Sałatki”, „Napoje” oraz „News”. Po otrzymaniu aktualnej listy z bazy danych dań rozpoczyna się generacja „karteczek” („karteczką” jest przedstawianie graficzne każdego poszczególnego dania w menu, zawierające zdjęcie, informację o nazwie, cenie, opisu oraz innych parametrach produktu), które są automatycznie dodawane do każdej z tych kategorii. Po wybraniu produktu użytkownik klika na przycisk „Dodaj”, po czym produkt trafia do koszyka. W pizzach przewidziany jest również wybór rozmiaru – przy jego zmianie automatycznie zmienia się cena. Również jest przewidziana możliwość edycji dodatków/składników w

prawym dolnym rogu karteczki, natomiast nie będą one załadowane na prezentacji ze względu na ograniczenia czasowe.

### Ekran\_koszyka

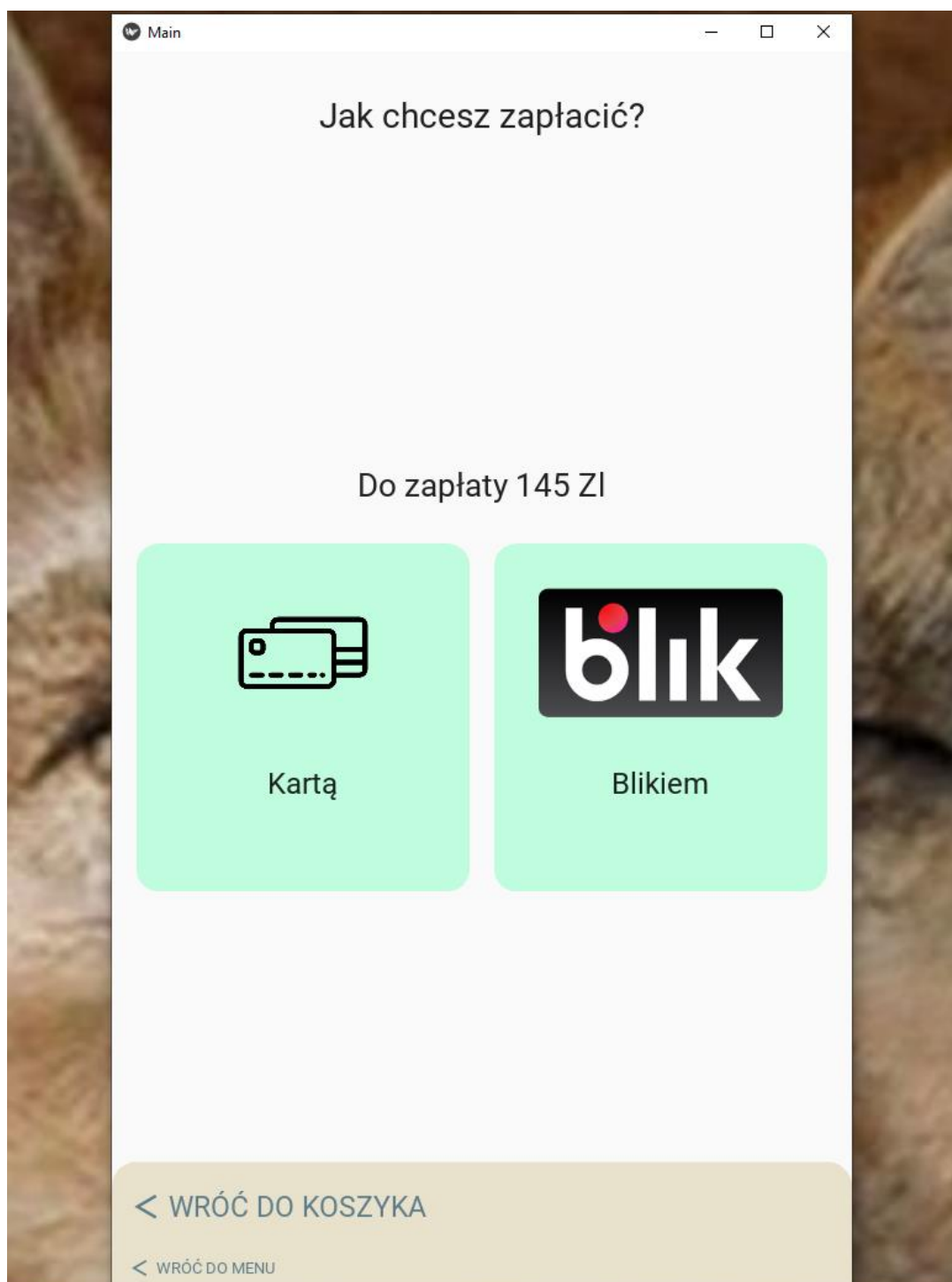


**Rysunek 32, Ekran koszyka.**

Źródło: Opracowanie własne.

W koszyku (rysunek 32) użytkownik ma możliwość usuwania pozycji lub zmniejszenia/zwiększenia ilości wybranych produktów. Również jest przewidziana możliwość powrotu do menu.

## Ekran\_wyboru\_płatności

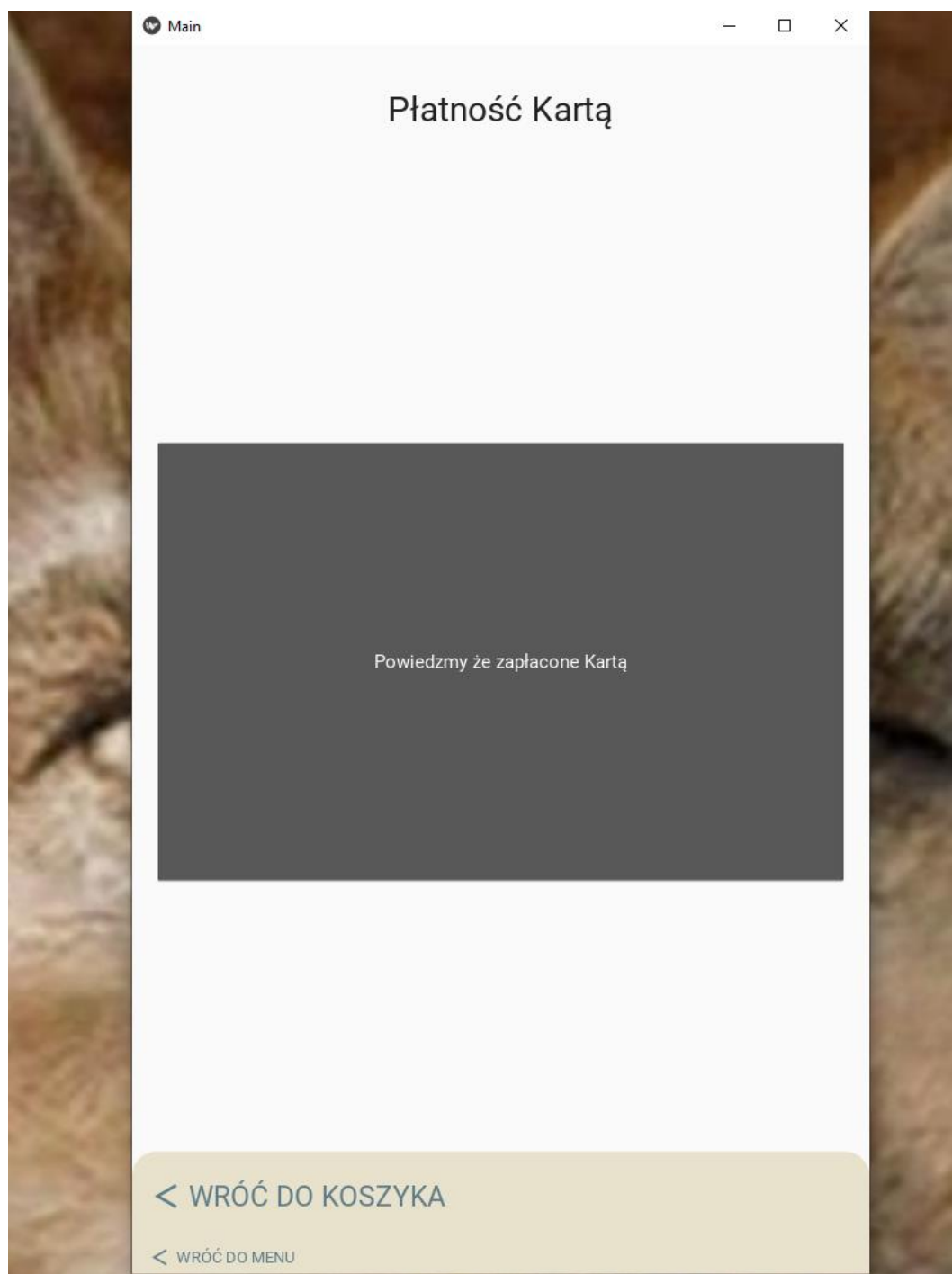


**Rysunek 33, Ekran wyboru płatności.**

Źródło: Opracowanie własne.

Na tym ekranie (rysunek 33) użytkownik decyduje czy będzie przeprowadzał płatność kartą czy Blikiem. Przewidziana jest możliwość cofnięcia do koszyka lub do menu z karteczkami.

## Ekran\_płatności

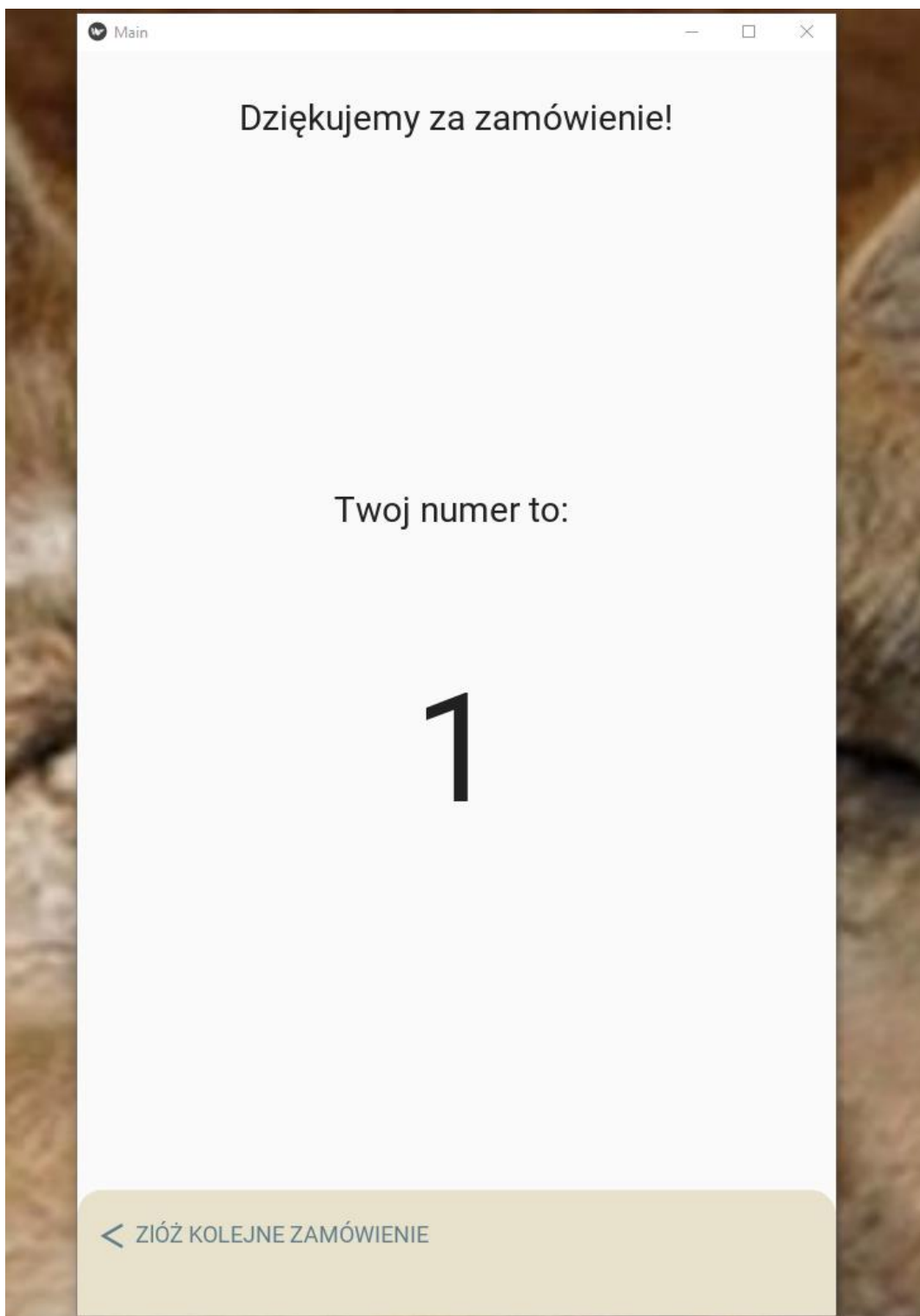


**Rysunek 34, Ekran płatności.**

Źródło: Opracowanie własne.

Ze względu na brak możliwości podłączenia terminalu płatniczego oraz trudnościami rejestracji działalności gospodarczej, w aplikacji jest realizowana symulacja płatności. Po naciśnięciu na przycisk (rysunek 34) aktywuje się funkcja, która ma być aktywowana tylko po udanej płatności.

## Ekran\_końcowy



**Rysunek 35, Ekran końcowy.**

Źródło: Opracowanie własne.

Na ekranie końcowym (rysunek 35) użytkownikowi pokazuje się numer jego zamówienia oraz podziękowanie, że zakupił u nas produkt. Przewidziana jest również możliwość cofnięcia do menu głównego, mające na celu złożenie kolejnego zamówienia. Podczas cofnięcia wyzerowana zostaje zawartość koszyka.



Po dokonaniu zakupu klient powinien otrzymać paragon z potwierdzeniem wykonanej transakcji, listę wszystkich wybranych artykułów, cenę końcową oraz numer, przydzielony do jego zamówienia. Do tego celu należy użyć wcześniej wymienioną drukarkę termiczną QR701, podłączoną do urządzenia.

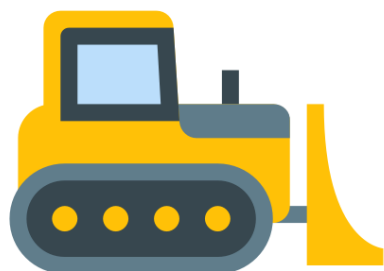
Drukarka termiczna podłączona jest poprzez USB-TTL przetwornicę i komunikuje się z aplikacją poprzez port szeregowy. Aplikacja ma do niego dostęp dzięki bibliotece Pyserial [29] w Pythonie, służącej do komunikacji z urządzeniami podłączonymi poprzez USB. Po symulacji transakcji włącza się kawałek kodu, odpowiadający za drukowanie paragonów. Część tego kodu, odpowiadająca za umiejscowienie informacji na drukarce jest przedstawiona niżej (rysunek 36) (załącznik 3).

```
542
543     ser.write('\x1d\x21\x22\x1b\x61\x01\n\nKusTech'.encode())
544     ser.write('\x1d\x21\x11\n\n*****'.encode())
545     ser.write('\x1d\x21\x04\nKaravaev Konstantin'.encode())
546     ser.write('\nnr.albumu: 303144'.encode())
547     ser.write('\nNIP: 521-40-15-102'.encode())
548     ser.write('\nPraca inzynierska PW WMT'.encode())
549     ser.write('\x1d\x21\x11\n\n*****'.encode())
550
551     ser.write(f'\x1b\x21\x01\x1b\x21\x03\n{year}-{month}-{day} {hour}-{minute}'.encode())
552     ser.write('\x1d\x21\x06\nParagon fiskalny\n\n'.encode())
553     ser.write('\x1b\x21\x01'.encode())
554     ser.write('\x1b\x21\x03'.encode())
555
556     print(WindowManager.koszyk_list)
557     # #Drukujemy menu
558     try:
559         a = WindowManager.koszyk_list[0][0]
560         for el in WindowManager.koszyk_list:
561             ser.write(f'{el[0]}\n'.encode())
562             ser.write(f'{el[4]}*{el[1].split()[0]}'.encode())
563             ser.write(f'\n{int(el[4])*int(el[1].split()[0])}.00 zł \n\n'.encode())
564     except:
565         for el in WindowManager.koszyk_list:
566             ser.write(f'{el[0]} {el[4]}*{el[1]} {int(el[4])*int(el[1].split()[0])} \n'.encode())
567     #Podsumowanie
568     ser.write('\x1d\x21\x11\n\n*****'.encode())
569     ser.write(f'\x1d\x21\x06\n---SUMA PLN: {WindowManager.cena_koncowa}----'.encode())
570     ser.write('\x1d\x21\x11\n\n*****\n\n'.encode())
571     ser.write('\x1d\x21\x06Numer twojego zamówienia:'.encode())
572     ser.write(f'\x1d\x21\x22\n\n{WindowManager.numer_zamowienia}'.encode())
573     ser.write('\x1d\x21\x11\n\n*****'.encode())
574     ser.write('\x1d\x21\x06\nDziekuje za zakup!\n\n\n'.encode())
```

**Rysunek 36, Część kodu, odpowiedzialna za odpowiednie rozmieszczenie informacji na paragonie.**  
Źródło: Opracowanie własne wzięte z kodu aplikacji.

## 5.4 Instalacja

W związku z tym, że komputer osoby projektującej funkcjonuje na systemie operacyjnym Windows, natomiast końcowe stanowisko samoobsługowe będzie działało na tablecie z systemem Android, należy przekształcić napisaną aplikację do odpowiedniego formatu – pliku o rozszerzeniu .apk. Do tego celu będę używał użyć Buildozer [30] (rysunek 37).



# buildozer

**Rysunek 37, Aplikacja do konwertacji w plik .apk.**

Źródło: Strona internetowa github.com: <https://github.com/IntracTo/buildozer> , data aktualizacji: 10.05.2023.

Buildozer to specjalistyczne narzędzie programistyczne, umożliwiające generowanie pakietów aplikacji Pythona w formacie binarnym, gotowych do uruchomienia na różnych platformach mobilnych, w tym na systemie Android. Poza jednym plikiem Buildozer pozwala na przekształcenie całych folderów z plikami, co w naszym przypadku jest potrzebne ze względu na użycie plików .py, .kv oraz kilku obrazków.

Do uruchomienia Buildozera lokalnie jest wymagany system operacyjny Linux, co jest pewnym utrudnieniem podczas projektowania. Istnieją trzy możliwości rozwiązania problemu – użycie odrębnego komputera mającego Linuxa, użycie subsystemu [31] oraz instalacja Linuxa jako wspomagający system na komputerze lub przekształcenie plików w chmurze Google Colab [32] (rysunek 38).

Projektant nie posiada oddzielnego komputera z systemem operacyjnym Linux, więc pierwsza opcja nie jest możliwa. Druga opcja z użyciem subsystemu na Linuxie nie jest najgorszym rozwiązaniem, natomiast wymaga dodatkowego czasu na instalację oraz pewne wymagania techniczne, które komputer projektanta nie jest w stanie spełnić. Wersja z użyciem Google Colaba wymaga tylko dostępu do Internetu oraz bazowych umiejętności współpracy z serwisami Google, dlatego wybrana została ta opcja.



**Rysunek 38, Google Colab.**

Źródło: Strona internetowa miteshparmar1.medium.com: <https://miteshparmar1.medium.com/structure-your-code-better-in-google-colab-with-text-and-code-cells-b6fa73feec20> , data aktualizacji: 10.05.2023.

Po przekonwertowaniu projektu aplikacji do jednego pliku .apk, został on przesłany na urządzenie z Androidem, po czym został zainstalowany na tablecie.



## 6. Obudowa

### 6.1 Wymagania

Wymagania, które powinna spełniać obudowa dla tworzonego systemu, są następujące:

1. Powinna być powtarzalna w produkcji.
2. Powinna wyglądać estetycznie.
3. Powinna być wielofunkcyjna.
4. Powinna mieć możliwość zamocowania na ścianie.
5. Nie powinna zajmować dużo miejsca.
6. Powinna mieć odpowiednią przestrzeń w środku na przechowywanie wszystkich kabli pomocniczych.

### 6.2 Technologia

Do produkcji korpusu postanowiono użyć metody przyrostowej druku 3D.

Technologia druku 3D polega na tworzeniu przedmiotów trójwymiarowych poprzez nakładanie materiału warstwa po warstwie. Podczas procesu drukowania, drukarka 3D przesuwa się nad stołem roboczym i nanosi materiał, tworząc kolejne warstwy. Materiałem tym może być na przykład tworzywo sztuczne, żywica, metal, beton czy nawet żywność. Drukarki 3D do każdego z wymienionych materiałów będą miały inne właściwości oraz technologię druku. Technologia druku 3D jest stosowana w wielu branżach, takich jak medycyna, przemysł lotniczy czy architektura.

Założyłem, że stworzę korpus z tworzywa – ale z jakiego jego rodzaju? Spośród wielu różnych odmian plastyków najpopularniejsze trzy rodzaje to są:

1. PLA (kwas polimlekowy) - ten biodegradowalny rodzaj plastiku jest jednym z najpopularniejszych materiałów do druku 3D, ze względu na swoją łatwość drukowania, szeroką dostępność, niską cenę oraz przyjazność dla środowiska. PLA dobrze się nadaje do drukowania różnych prototypów lub elementów dekoracyjnych.
2. ABS (akrylonitryladienobutaństyren) - ten materiał jest zdecydowanie trudniejszy w drukowaniu w porównaniu do druku za pomocą PLA, natomiast wciąż jest jednym z najpopularniejszych w swojej branży. ABS jest stosowany do drukowania obiektów wymagających wytrzymałości mechanicznej, takich jak elementy maszynowe, obudowy czy zabawki.
3. PETG (polietylenotereftalan etylenu z glikolem) - ten materiał coraz częściej zyskuje na popularności, ze względu na swoją wytrzymałość mechaniczną, odporność na uderzenia oraz elastyczność. PETG jest często stosowany do drukowania obiektów, które wymagają trwałości i wytrzymałości, takich jak elementy maszynowe czy narzędzia.



**Rysunek 39, Szpule plastyku PLA.**

Źródło: Opracowanie własne.

W związku z tym, że powstające urządzenie jest prototypem, postanowiono użyć tworzywa z pierwszego punktu – tworzywa PLA.

**Tabela 1, Wybrane parametry plastyku PLA.**

Źródło: <https://global3d.pl/pl/blog/filament-pla-wytrzymalosc-temperatura-i-zalety-b32.html>

Właściwość	Wartość
Pełna nazwa	<i>Kwas polimlekowy</i>
Wytrzymałość na rozciąganie	<i>37 MPa</i>
Rozciąganie	<i>6%</i>
Moduł giętkości	<i>4GPa</i>
Gęstość	<i>1,3 <math>\frac{g}{cm^3}</math></i>
Temperatura topnienia	<i>173°C</i>
Temperatura zeszklenia	<i>60°C</i>

Do jego przetopienia oraz formułowania gotowego wyrobu zostanie użyta drukarka 3D (rysunek 9), złożona z części customowych, pochodzących z różnych źródeł oraz krajów,

więc nieposiadająca ustalonego producenta oraz uporządkowanej dokumentacji technicznej. Dostępna drukarka składa się ze stołu, dyszy, silników krokowych z prowadnicami oraz sterownikiem (na bazie Arduino).

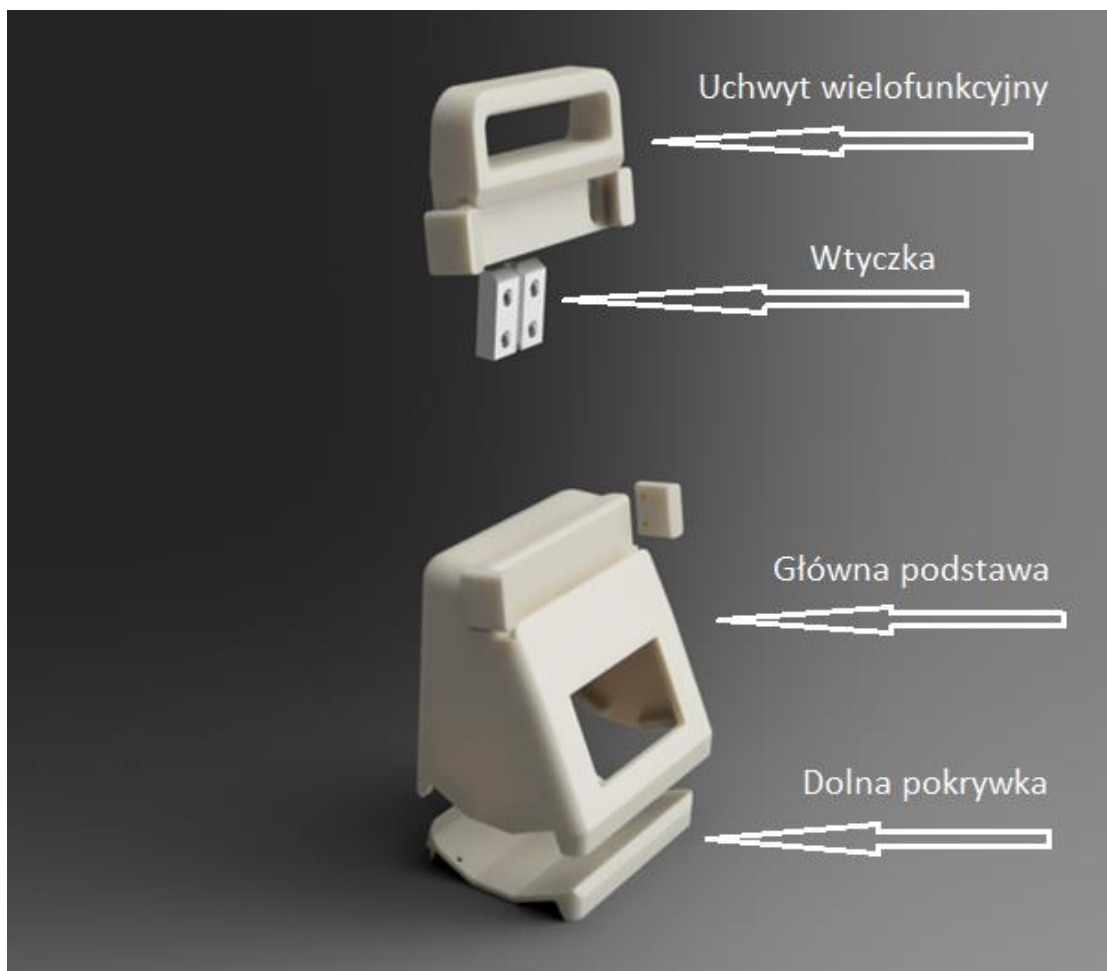
W przypadku drukowania korpusu z tworzywa PLA proces będzie wyglądał następująco: na początku drukarka podgrzeje tworzywo do odpowiedniej temperatury, a następnie zacznie nanosić pierwszą warstwę na stół. Po ukończeniu pierwszej warstwy nagrzana dysza podniesie się w górę o drobinę, po czym drukarka nałoży kolejną warstwę na poprzednią. Proces ten będzie trwał, aż do uzyskania pełnej gotowości elementu drukowanego.

Korpus, który należy wydrukować, będzie składał się z wielu warstw o grubości mniejszej od milimetra. Każda kolejna warstwa będzie nakładana na poprzednią, aż do uzyskania końcowego kształtu figury.

Dużą zaletą druku 3D jest to, że w trakcie drukowania będzie istniała możliwość kontroli nałożenia każdej warstwy, co zwiększa przejrzystość procesu produkcyjnego.

### **6.3 Proces projektowy**

Z racji tego, że dostępna drukarka 3D nie posiada ogromnego pola roboczego, wymaganego dla wytworzenia całego korpusu na raz, została podjęta decyzja o podzieleniu korpusu (rysunek 40) na dwie części – górną (rysunek 41) i dolną (rysunek 42).



**Rysunek 40, Projekt obudowy stanowiska samoobsługowego.**

Źródło: Opracowanie własne.

Postanowiono, że górna część korpusu będzie zawierała uchwyt wielofunkcyjny, oraz gniazdo na wtyczkę z tyłu, która powinna być przymocowana do powierzchni pionowej. Układ gniazdo-wtyczka będzie potrzebny na wypadek, gdy właściciel placówki postanowi zamocować system na ścianie swojej placówki.

Uchwyt wielofunkcyjny posiada wcięcie do nałożenia na tablet, rączkę do wygodnego trzymania stanowiska na wypadek jego przeniesienia oraz otwory na śruby stabilizujące do szcęk.

Wtyczka posiada 4 otwory na śruby mające na celu umożliwić jej przekręcenie do powierzchni.



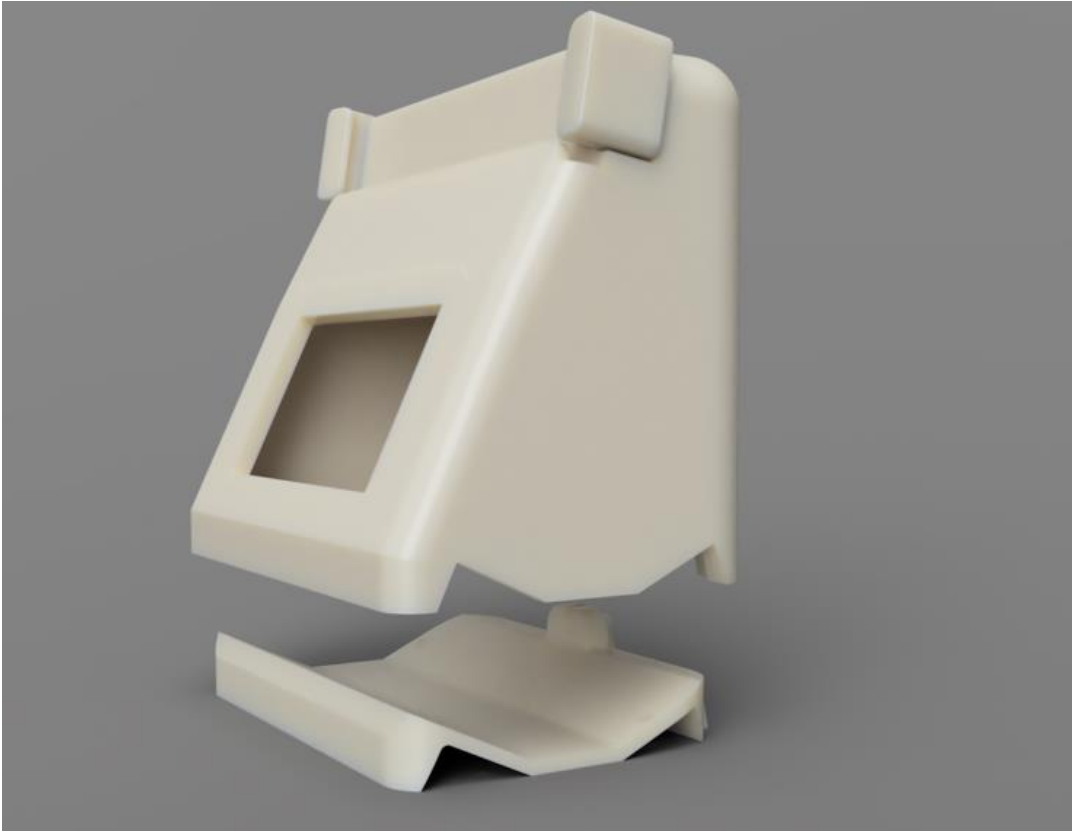
**Rysunek 41, Górna część obudowy - uchwyt wielofunkcyjny.**

Źródło: Opracowanie własne.

Dolna część zaprojektowanego korpusu będzie zawierała miejsce na wszystkie przewody oraz dodatkowe elementy, potrzebne do działania systemu. Również będzie w niej rozmieszczona drukarka termiczna.

Główna podstawa posiada wcięcie do postawienia tabletu, wcięcie na kabel zasilający, otwór do podłączenia tabletu wtyczką USB-C, otwory stabilizujące do szczęk oraz 4 otwory do zamocowania dolnej pokrywki

Dolna pokrywka posiada miejsce na 4 otwory mocujące ją do głównej podstawy oraz jeden otwór na kabel zasilający.



**Rysunek 42, Dolna część obudowy - podstawa konstrukcji.**

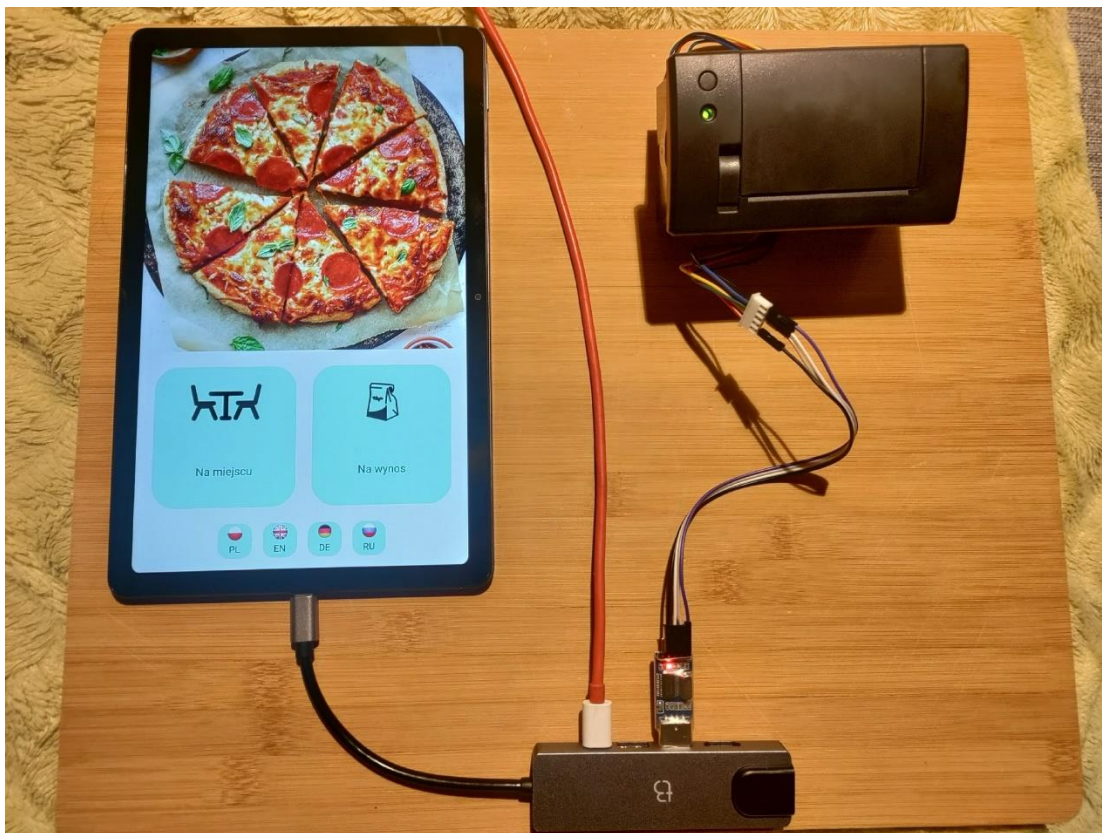
Źródło: Opracowanie własne.

Pomiędzy górną a dolną częściami obudowy będzie umieszczony tablet, natomiast jak można zapełnić mu stabilną pozycję? Do tego celu zostały zaprojektowane szczęki ze śrubami na obu częściach korpusu. Po nałożeniu obu części na tablet osoba montująca dokręca śruby mocujące, tym samym zapewnia mu dobre połączenie mechaniczne z obudową.

## 7. Testowanie

Po przygotowaniu poszczególnych części systemu należy je złożyć. Po rozmieszczeniu aplikacji webowej na serwerach AWS zostało wysłane pierwsze zapytanie do bazy danych wtedy, gdy jeszcze aplikacja nie była na komputerze – zakończyło się ono powodzeniem. Następnie aplikacja została przebudowana na plik .apk oraz zainstalowana na tablecie. Po zainstalowaniu aplikacji przy próbie jej pierwszego włączenia równo po dwóch sekundach ładowania automatycznie się wyłączała, nawet nie pokazując ekranu witającego użytkownika. Po wielu próbach naprawy istniejącej sytuacji postanowiono skorzystać z mostu ADB (Android Debug Bridge) [33], dzięki którego zastosowaniu udało się wykryć, że niektóre biblioteki (głównie biblioteka „requests” [34]) użyte podczas projektowania logiki działania aplikacji wymagają zainstalowania dodatkowych modułów dla prawidłowego ich funkcjonowania. Po ich instalacji problem się rozwiązał i aplikacja się włączyła na tablecie.

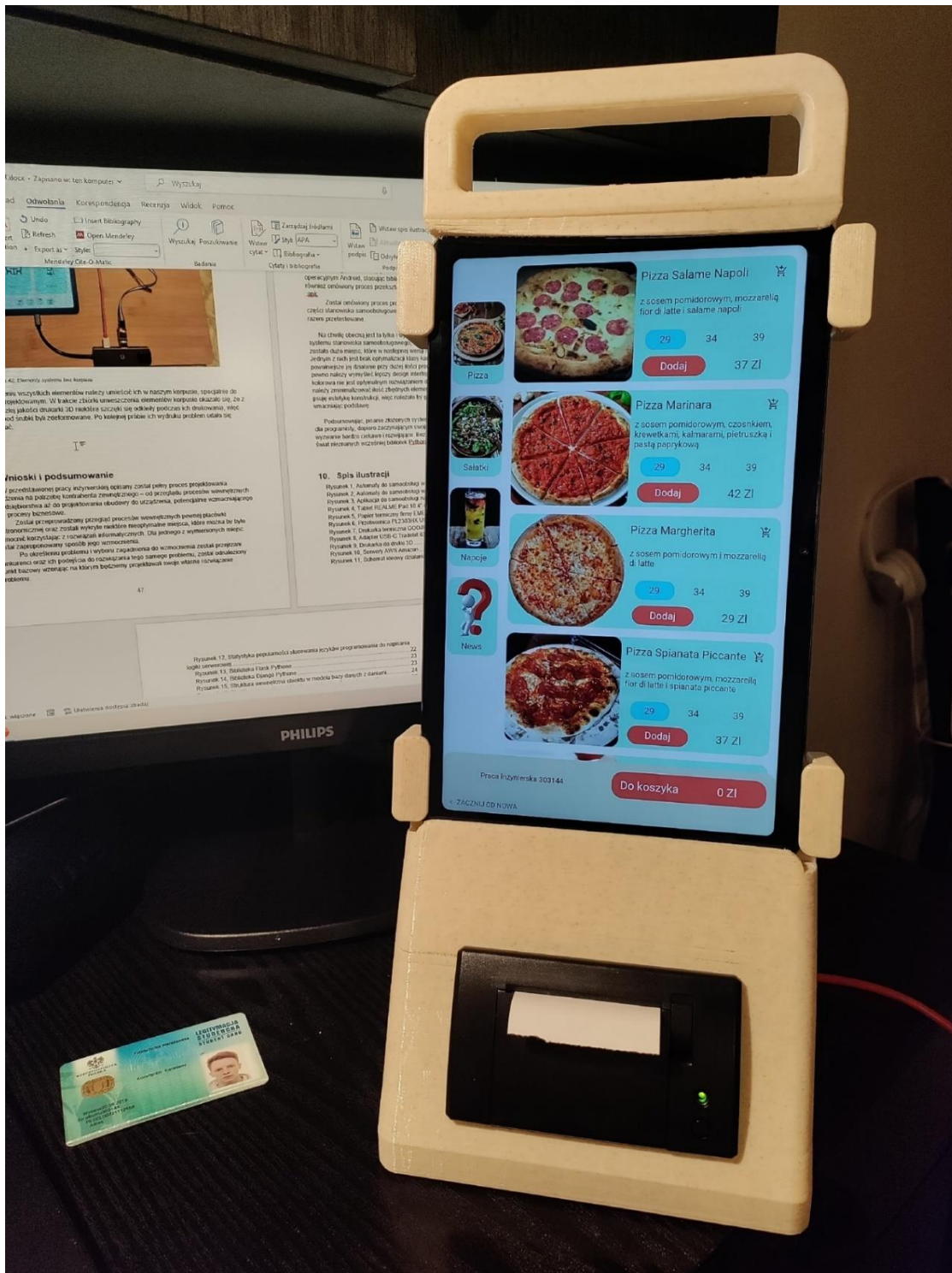
Następnie należało podłączyć drukarkę termiczną do tabletu – był to ogromny problem. Podłączając ją do komputera z Windowsem, wszystko działało bez problemu – na Windowsie istnieje możliwość dostępu do portów szeregowych USB oraz do urządzeń, które są przez nie połączone. Natomiast na tablecie gniazdo USB jest tylko jedno oraz dostęp do niego użytkownikowi jest utrudniony. Przez długi czas było wykonane wiele prób otrzymania do niego dostępu – na chwilę napisania tekstu problem wciąż nie został rozwiązany (rysunek 43).



**Rysunek 43, Elementy systemu bez korpusu.**  
Źródło: Opracowanie własne.

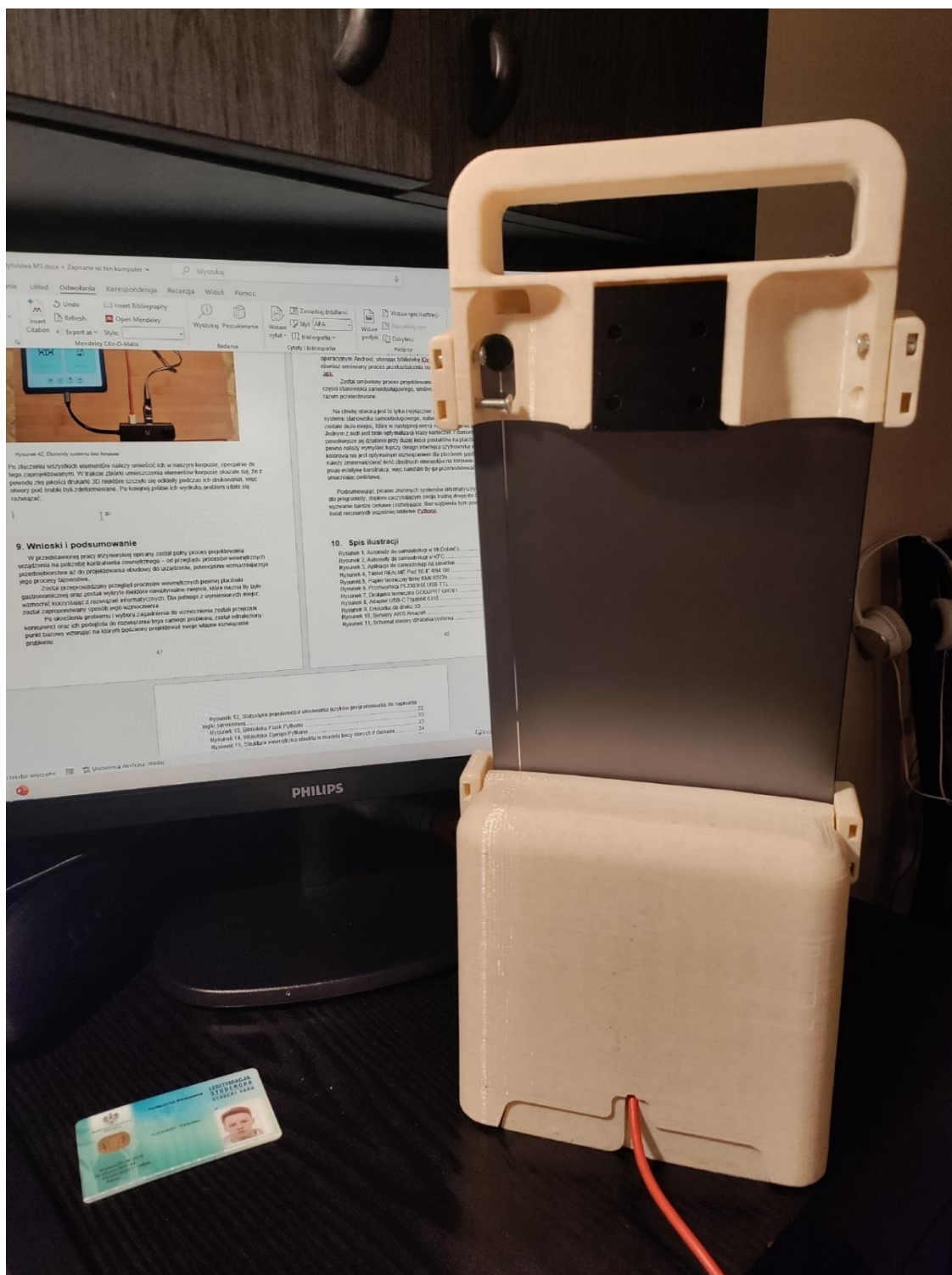


Po złączeniu wszystkich elementów należy umieścić je w korpusie, specjalnie do tego zaprojektowanym. W trakcie umieszczenia elementów w korpusie okazało się, że z powodu nieodpowiedniej jakości drukarki 3D niektóre szczegóły się odkleiły podczas ich drukowania, w związku z czym otwory pod śrubki były zdeformowane. Po kolejnej próbie ich wydruku problem udało się rozwiązać (rysunki 44 i 45),.



**Rysunek 44, Elementy systemu z korpusem - widok z przodu.**  
Źródło: Opracowanie własne.





**Rysunek 45, Elementy systemu z korpusem - widok od tyłu.**  
 Źródło: Opracowanie własne.

## 8. Wnioski i podsumowanie

W przedstawionej pracy inżynierskiej opisany został pełny proces projektowania urządzenia na potrzebę kontrahenta zewnętrznego – od przeglądu procesów wewnętrznych przedsiębiorstwa aż do projektowania obudowy do urządzenia, potencjalnie wzmacniającego jego procesy biznesowe.

Został przeprowadzony przegląd procesów wewnętrznych pewnej placówki gastronomicznej oraz zostały wykryte niektóre nieoptymalne miejsca, które można by było wzmocnić, korzystając z rozwiązań informatycznych. Dla jednego z wymienionych miejsc został zaproponowany sposób jego wzmocnienia.

Po określeniu problemu i wyboru zagadnienia do wzmocnienia zostali przejrzeni konkurenci oraz ich podejścia do rozwiązania tego samego problemu, został odnaleziony punkt bazowy, wzorując na którym będziemy projektowali swoje własne rozwiązanie problemu.

W pracy również został przedstawiony proces wykrycia wytycznych początkowych systemu, dobór sprzętu potrzebnego do jego realizacji oraz projektowanie architektury ideowej działania systemu. Został ten proces przedstawiony w formie logicznego ciągu myśli osoby projektującej, wspieranej obecną sytuacją na rynku.

Po ustaleniu ogólnej koncepcji systemu przedstawiony został proces projektowania aplikacji serwerowej z bazą danych dań wymienionej placówki, opisany został proces współpracy z bazą danych, korzystając z biblioteki Django języka Python, pracy z jego panelem administracyjnym oraz z możliwością autoryzacji stosując tokeny podczas komunikacji za pomocą protokołu API.

Został przedstawiony proces projektowy aplikacji na urządzenie z systemem operacyjnym Android, stosując bibliotekę Kivy oraz KivyMD języka Python, po czym został również omówiony proces przekształcenia napisanych plików do odpowiedniego formatu .apk.

Został omówiony proces projektowania obudowy do druku 3D oraz wszystkie części stanowiska samoobsługowego, omówione wyżej, zostały złożone do jednego systemu i razem przetestowane.

Na chwilę obecną jest to tylko i wyłącznie prototyp przyszłościowego komercyjnego systemu stanowiska samoobsługowego, natomiast podczas jego projektowania została wykryta duża liczba miejsc, które w następnej wersji na pewno będą poprawione i polepszone.

Jednym z nich jest brak optymalizacji klasy karteczek z daniami w aplikacji, co powoduje powolniejsze jej działanie przy dużej ilości produktów w placówce. Druga kwestia – na pewno należy wymyślić lepszy design interfejsu użytkownika w aplikacji. Wybrana paleta kolorowa nie jest optymalnym rozwiązaniem dla placówek gastronomicznych. Trzeci punkt – należy zminimalizować liczbę zbędnych elementów w korpusie. Uchwyt wielofunkcyjny tylko psuje estetykę konstrukcji, więc należałoby go przemodelować lub w całości usunąć, umacniając podstawę.

Wprowadzenie podobnych systemów ma duży potencjał nie tylko w branży gastronomicznej, ale praktycznie w każdej dziedzinie współczesnej gospodarki. Automatyzacja zwiększa prędkość procesów wewnętrznych firmy, co potencjalnie skutkuje większą przepustowością firmy, z czego wynikają większe przychody oraz dominacja nad firmami trzymającymi się daleko od skomplikowanych technologii.

Podsumowując, pisanie złożonych systemów informatycznych jest trudnym wyzwaniem dla programisty, który dopiero zaczyna swoją trudną drogę do Seniora, natomiast jest to wyzwanie bardzo ciekawe i rozwijające. Bez wątpienia bym powtórzył tę ciekawą podróż w świat nieznanych wcześniej bibliotek Pythona.

## 9. Bibliografia

1. Strona internetowa niemieckiego producenta automatów samoobsługowych  
[https://www.dieboldnixdorf.com/-/media/diebold/ag-downloads/poslotterysystems/manuals/kiosk/w1000\\_usermanual.pdf](https://www.dieboldnixdorf.com/-/media/diebold/ag-downloads/poslotterysystems/manuals/kiosk/w1000_usermanual.pdf)  
stan na 10.05.2023
2. Strona internetowa MediaExpert z dostępną na rynku ofertą tabletów  
<https://www.mediaexpert.pl/komputery-i-tablety/tablety-i-e-booki/tablety>  
stan na 10.05.2023
3. Strona internetowa producenta Realme z parametrami urządzenia  
<https://www.realme.com/global/realme-pad/specs>  
stan na 10.05.2023
4. Strona internetowa z wybranym papierem termicznym  
<https://www.elpro.com.pl/emerson/rolki-termiczne-57-20-10szt/927.html>  
stan na 10.05.2023
5. Strona internetowa z dokumentacją techniczną układu PL2303HX  
<https://www.alldatasheet.com/datasheet-pdf/pdf/1179018/PROLIFIC/PL2303HX.html>  
stan na 10.05.2023
6. Strona internetowa z dokumentacją techniczną drukarki **GOOJPRT QR701**  
<https://cdn-shop.adafruit.com/datasheets/CSN-A2+User+Manual.pdf>  
stan na 10.05.2023
7. Strona internetowa z informacją o opłatach za terminale płatnicze  
<https://mico.pl/baza-wiedzy/oplaty-za-terminale-platnicze-2023/>  
stan na 10.05.2023
8. Strona internetowa z adapterem USB-C Tradebit 6318  
<https://parts-store.pl/product-pol-2197-Adapter-przejsciowka-HUB-USB-C-HDMI-4k-USB-3-0-PD.html>  
stan na 10.05.2023
9. Grzesiak W., Niesłony P., Kiszka P., Programowanie obrabiarek CNC, Wydawnictwo Naukowe PWN , Warszawa 2020
10. Горьков Д., Холмогоров В., 3D-печать с нуля, Издательство БХВ-Петербург, Санкт-Петербург 2020
11. Strona internetowa z informacją systemach POS  
<https://www.oracle.com/pl/retail/pos-systems/what-is-retail-pos/>  
stan na 10.05.2023
12. Strona internetowa AWS  
[https://aws.amazon.com/?nc2=h\\_lg](https://aws.amazon.com/?nc2=h_lg)  
stan na 10.05.2023
13. Strona internetowa Visual Studio Code  
<https://code.visualstudio.com/>  
stan na 10.05.2023
14. Strona internetowa z informacją protokole API  
<https://support.apple.com/pl-pl/guide/shortcuts-mac/apd2e30c9d45/mac>  
stan na 10.05.2023
15. Danjou J., Python na poważnie, Wydawnictwo Naukowe PWN , Warszawa 2019
16. Bird A., Guest C., Badhwar S., Shaw B., Chandra Bharth K S, Django. Tworzenie nowoczesnych aplikacji internetowych w pythonie, Wydawnictwo Helion, Gliwice 2022

17. Strona internetowa Python Flask  
<https://flask.palletsprojects.com/en/2.3.x/>  
stan na 10.05.2023
18. Strona internetowa z dokumentacją panelu administracyjnego Python Django  
<http://ktm.umg.edu.pl/~pik/prokom/pliki/njitp-l11.pdf>  
stan na 10.05.2023
19. Strona internetowa z dokumentacją o autoryzacji użytkowników Python Django  
<https://docs.djangoproject.com/en/4.2/topics/auth/>  
stan na 10.05.2023
20. Strona internetowa z dokumentacją o autoryzacji tokenem Python Django Rest API  
<https://www.django-rest-framework.org/api-guide/authentication/>  
stan na 10.05.2023
21. Strona internetowa z dokumentacją Python Django Rest API  
<https://www.django-rest-framework.org/>  
stan na 10.05.2023
22. Strona internetowa z kosztem użytego serwera AWS  
[https://aws.amazon.com/lightsail/pricing/?trk=be4c6119-fb07-4f51-9f48-21e2c72b3d51&sc\\_channel=ps&ef\\_id=Cj0KCQjwpPKiBhDvARIsACn-gzDKJsC7QAslz8cSoKXQj2h4FHp2q5N\\_gSSabMpXgSa3zzchk-oSYnlaAISeEALw\\_wcB:G:s&s\\_kwcid=AL!4422!3!645133581981!e!!g!!aws%20lightsail%20pricing!19579657655!148952120567](https://aws.amazon.com/lightsail/pricing/?trk=be4c6119-fb07-4f51-9f48-21e2c72b3d51&sc_channel=ps&ef_id=Cj0KCQjwpPKiBhDvARIsACn-gzDKJsC7QAslz8cSoKXQj2h4FHp2q5N_gSSabMpXgSa3zzchk-oSYnlaAISeEALw_wcB:G:s&s_kwcid=AL!4422!3!645133581981!e!!g!!aws%20lightsail%20pricing!19579657655!148952120567)  
stan na 10.05.2023
23. Strona internetowa Ubuntu  
<https://ubuntu.com/>  
stan na 10.05.2023
24. Strona internetowa z informacją o SSH  
<https://www.ssh.com/academy/ssh/protocol>  
stan na 10.05.2023
25. Cywiak D., Cywiak M., Multi-Platform Graphics Programming with Kivy, Wydawnictwo APress, 2021
26. Strona internetowa z dokumentacją Python KivyMD  
<https://kivymd.readthedocs.io/en/1.1.1/>  
stan na 10.05.2023
27. Strona internetowa z dokumentacją języka Kv  
<https://kivy.org/doc/stable/guide/lang.html>  
stan na 10.05.2023
28. Strona internetowa z dokumentacją JSON dla pracy z Pythonem  
<https://docs.python.org/3/library/json.html>  
stan na 10.05.2023
29. Strona internetowa z dokumentacją biblioteki Pyserial  
<https://pyserial.readthedocs.io/en/latest/shortintro.html>  
stan na 10.05.2023
30. Strona internetowa z dokumentacją Buildozer dla Kivy  
<https://kivy.org/doc/stable/guide/packaging-android.html>  
stan na 10.05.2023
31. Strona internetowa z opisem instalacji Linuxa stosując subsystem WSL  
<https://learn.microsoft.com/en-us/windows/wsl/install>  
stan na 10.05.2023

32. Środowisko Google Colab do przekształcenia plików kivy na .apk  
<https://learn.microsoft.com/en-us/windows/wsl/install>  
stan na 10.05.2023
33. Strona internetowa z dokumentacją Android Debug Bridge (ADB)  
<https://developer.android.com/tools/adb>  
stan na 10.05.2023
34. Strona internetowa z dokumentacją biblioteki Python requests  
<https://requests.readthedocs.io/en/latest/>  
stan na 10.05.2023

## 10. Spis rysunków

<b>Rysunek 1, Automaty do samoobsługi w McDolald`s.</b> Źródło: Strona internetowa tapbox.eu : <a href="https://tapbox.eu/more-money-is-spent-using-self-service-kiosks/">https://tapbox.eu/more-money-is-spent-using-self-service-kiosks/</a> , data aktualizacji: 10.05.2023. ....	12
<b>Rysunek 2, Automaty do samoobsługi w KFC.</b> Źródło: Strona internetowa kiosksoft.ru : <a href="https://kiosksoft.ru/news/2020/06/11/kfc-zapuskaet-polnostyu-avtomatizirovannyj-restoran-v-moskve-68647">https://kiosksoft.ru/news/2020/06/11/kfc-zapuskaet-polnostyu-avtomatizirovannyj-restoran-v-moskve-68647</a> , data aktualizacji: 10.05.2023.....	13
<b>Rysunek 3, Przykład aplikacja do samoobsługi na smartfon.</b> Źródło: Strona internetowa dribbble.com: <a href="https://dribbble.com/shots/8724373-Fast-Food-App-Design">https://dribbble.com/shots/8724373-Fast-Food-App-Design</a> , data aktualizacji: 10.05.2023. ....	14
<b>Rysunek 4, Tablet REALME Pad 10.4" 4/64 GB.</b> Źródło: Strona internetowa mediamarkt.pl: <a href="https://mediamarkt.pl/komputery-i-tablety/tablet-realme-pad-10-4-2022-wifi-4gb-64gb-szary?gclid=CjwKCAjwge2iBhBBEiwAfXDBRz9bcPuZTwXOn93wogCYAcnBo-m0ck_alELO_FgUtElgWX8asPjwERoCyjgQAvD_BwE&amp;gclidsrc=aw.ds">https://mediamarkt.pl/komputery-i-tablety/tablet-realme-pad-10-4-2022-wifi-4gb-64gb-szary?gclid=CjwKCAjwge2iBhBBEiwAfXDBRz9bcPuZTwXOn93wogCYAcnBo-m0ck_alELO_FgUtElgWX8asPjwERoCyjgQAvD_BwE&amp;gclidsrc=aw.ds</a> , data aktualizacji: 10.05.2023.....	16
<b>Rysunek 5, Papier termiczny firmy EMERSON.</b> Źródło: Strona internetowa allegro.pl: <a href="https://allegro.pl/oferta/rolka-kasowa-termiczna-57-20m-10szt-emerson-6711845882">https://allegro.pl/oferta/rolka-kasowa-termiczna-57-20m-10szt-emerson-6711845882</a> , data aktualizacji: 10.05.2023. ....	17
<b>Rysunek 6, Przetwornica PL2303HX USB-TTL.</b> Źródło: Strona internetowa sklep.msalamon.pl: <a href="https://sklep.msalamon.pl/produkt/konwerter-usb-rs232-pl2303-ttl/">https://sklep.msalamon.pl/produkt/konwerter-usb-rs232-pl2303-ttl/</a> , data aktualizacji: 10.05.2023. ....	18
<b>Rysunek 7, Drukarka termiczna GOOJPRT QR701.</b> Źródło: Strona internetowa pl.aliexpress.com: <a href="https://pl.aliexpress.com/item/4000089296652.html">https://pl.aliexpress.com/item/4000089296652.html</a> , data aktualizacji: 10.05.2023.....	19
<b>Rysunek 8, Adapter USB-C Tradebit 6318.</b> Źródło: Strona internetowa ceneo.pl: <a href="https://www.ceneo.pl/137069814">https://www.ceneo.pl/137069814</a> , data aktualizacji: 10.05.2023. ....	20
<b>Rysunek 9, Drukarka do druku 3D.</b> Źródło: Opracowanie własne – sprzęt dostępny projektantowi na czas przygotowania pracy inżynierskiej.....	21
<b>Rysunek 10, Serwery AWS Amazon.</b> Źródło: Strona internetowa excelonsolutions.com: <a href="https://www.excelonsolutions.com/aws-services/">https://www.excelonsolutions.com/aws-services/</a> , data aktualizacji: 10.05.2023. ....	22
<b>Rysunek 11, Schemat ideowy działania systemu.</b> Źródło: Opracowanie własne stworzone na podstawie opisanej wyżej koncepcji systemu.....	23
<b>Rysunek 12, Statystyka popularności stosowania języków programowania do napisania logiki serwerowej.</b> Źródło: Strona internetowa blog.back4app.com: <a href="https://blog.back4app.com/backend-development-languages/">https://blog.back4app.com/backend-development-languages/</a> , data aktualizacji: 10.05.2023. ....	25
<b>Rysunek 13, Biblioteka Flask Pythone.</b> Źródło: Strona internetowa pl.m.wikipedia.org: <a href="https://pl.m.wikipedia.org/wiki/Plik:Flask_logo.svg">https://pl.m.wikipedia.org/wiki/Plik:Flask_logo.svg</a> , data aktualizacji: 10.05.2023.....	26
<b>Rysunek 14, Biblioteka Django Pythone.</b> Źródło: Strona internetowa www.djangoproject.com: <a href="https://www.djangoproject.com/community/logos/">https://www.djangoproject.com/community/logos/</a> , data aktualizacji: 10.05.2023.....	26
<b>Rysunek 15, Struktura wewnętrzna obiektu w modelu bazy danych z daniami.</b> Źródło: Opracowanie własne.....	27
<b>Rysunek 16, Struktura wewnętrzna obiektu modelu z konfiguracją początkową.</b> Źródło: Opracowanie własne.....	29

<b>Rysunek 17, Logowanie do panelu administracyjnego na Django.</b> Źródło: Opracowanie własne znajduje się pod adresem <a href="http://3.123.205.81:8000/admin/">http://3.123.205.81:8000/admin/</a> , data aktualizacji: 10.05.2023.....	30
<b>Rysunek 18, Panel administracyjny na Django.</b> Źródło: Opracowanie własne znajduje się pod adresem <a href="http://3.123.205.81:8000/admin/">http://3.123.205.81:8000/admin/</a> , data aktualizacji: 10.05.2023. ....	30
<b>Rysunek 19, Zawartość modelu bazy danych z daniami.</b> Źródło: Opracowanie własne znajduje się pod adresem <a href="http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/">http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/</a> , data aktualizacji: 10.05.2023.....	31
<b>Rysunek 20, Zawartość obiektu z modelu bazy danych z daniami (1).</b> Źródło: Opracowanie własne znajduje się pod adresem <a href="http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/3/change/">http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/3/change/</a> , data aktualizacji: 10.05.2023.....	32
<b>Rysunek 21, Zawartość obiektu z modelu bazy danych z daniami (2).</b> Źródło: Opracowanie własne znajduje się pod adresem <a href="http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/3/change/">http://3.123.205.81:8000/admin/pizzeria/dania_pizzeria/3/change/</a> , data aktualizacji: 10.05.2023.....	32
<b>Rysunek 22, Schemat działania protokołu API.</b> Źródło: Strona internetowa <a href="https://www.mindinventory.com/blog/best-practices-rest-api-development/">mindinventory.com: https://www.mindinventory.com/blog/best-practices-rest-api-development/</a> , data aktualizacji: 10.05.2023.....	33
<b>Rysunek 23, Wysyłanie zapytania na serwer z poziomu aplikacji.</b> Źródło: Opracowanie własne.....	33
<b>Rysunek 24, Moduły potrzebne do działania aplikacji serwerowej.</b> Źródło: Opracowanie własne.....	34
<b>Rysunek 25, Szafy serwerowe.</b> Źródło: Strona internetowa <a href="https://bestmicro.pl/blog/15_serwery-co-warto-o-nich-wiedziec.html">bestmicro.pl: https://bestmicro.pl/blog/15_serwery-co-warto-o-nich-wiedziec.html</a> , data aktualizacji: 10.05.2023.....	35
<b>Rysunek 26, Wykupiony serwer AWS do funkcjonowania aplikacji webowej.</b> Źródło: Opracowanie własne bazujące na stronie <a href="https://lightsail.aws.amazon.com/ls/webapp/eu-central-1/instances/Django-1/connect">lightsail.aws.amazon.com: https://lightsail.aws.amazon.com/ls/webapp/eu-central-1/instances/Django-1/connect</a> , data aktualizacji: 10.05.2023.....	35
<b>Rysunek 27, Biblioteka Kivy Python.</b> Źródło: Strona internetowa <a href="https://lowendplay.com/python-kivy/">lowendplay.com: https://lowendplay.com/python-kivy/</a> , data aktualizacji: 10.05.2023. ....	37
<b>Rysunek 28, Biblioteka KivyMD Python.</b> Źródło: Strona internetowa <a href="https://repository-images.githubusercontent.com/284716598/5af8e880-d8ba-11ea-9ce1-e5e8d603143f">github.com: https://repository-images.githubusercontent.com/284716598/5af8e880-d8ba-11ea-9ce1-e5e8d603143f</a> , data aktualizacji: 10.05.2023. ....	38
<b>Rysunek 29, Ekran witający użytkownika.</b> Źródło: Opracowanie własne. ....	39
<b>Rysunek 30, Ekran wyboru miejsca.</b> Źródło: Opracowanie własne. ....	40
<b>Rysunek 31, Ekran menu.</b> Źródło: Opracowanie własne. ....	41
<b>Rysunek 32, Ekran koszyka.</b> Źródło: Opracowanie własne. ....	42
<b>Rysunek 33, Ekran wyboru płatności.</b> Źródło: Opracowanie własne.....	43
<b>Rysunek 34, Ekran płatności.</b> Źródło: Opracowanie własne. ....	44
<b>Rysunek 35, Ekran końcowy.</b> Źródło: Opracowanie własne.....	45
<b>Rysunek 36, Część kodu, odpowiedzialna za odpowiednie rozmieszczenie informacji na paragonie.</b> Źródło: Opracowanie własne wzięte z kodu aplikacji. ....	46
<b>Rysunek 37, Aplikacja do konwertacji w plik .apk.</b> Źródło: Strona internetowa <a href="https://github.com/Intracto/buildozer">github.com: https://github.com/Intracto/buildozer</a> , data aktualizacji: 10.05.2023.....	47



<b>Rysunek 38, Google Colab.</b> Źródło: Strona internetowa miteshparmar1.medium.com: <a href="https://miteshparmar1.medium.com/structure-your-code-better-in-google-colab-with-text-and-code-cells-b6fa73feec20">https://miteshparmar1.medium.com/structure-your-code-better-in-google-colab-with-text-and-code-cells-b6fa73feec20</a> , data aktualizacji: 10.05.2023. ....	48
<b>Rysunek 39, Szpule plastyku PLA.</b> Źródło: Opracowanie własne. ....	50
<b>Rysunek 40, Projekt obudowy stanowiska samoobsługowego.</b> Źródło: Opracowanie własne. ....	52
<b>Rysunek 41, Górna część obudowy - uchwyt wielofunkcyjny.</b> Źródło: Opracowanie własne. ....	53
<b>Rysunek 42, Dolna część obudowy - podstawa konstrukcji.</b> Źródło: Opracowanie własne. ....	54
<b>Rysunek 43, Elementy systemu bez korpusu.</b> Źródło: Opracowanie własne. ....	55
<b>Rysunek 44, Elementy systemu z korpusem - widok z przodu.</b> Źródło: Opracowanie własne. ....	56
<b>Rysunek 45, Elementy systemu z korpusem - widok od tyłu.</b> Źródło: Opracowanie własne. ....	57

## 11. Spis tabel

<b>Tabela 1, Wybrane parametry plastyku PLA.</b> Źródło: <a href="https://global3d.pl/pl/blog/filament-pla-wytrzymalosc-temperatura-i-zalety-b32.html">https://global3d.pl/pl/blog/filament-pla-wytrzymalosc-temperatura-i-zalety-b32.html</a> .....	50
--	----

## 12 Załączniki.

### Załącznik 1, Plik „models.py” aplikacji webowej.

```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.

class Dania_Pizzeria(models.Model):
    Id = models.AutoField(primary_key=True)
    Kategoria = models.ForeignKey('Category', on_delete=models.PROTECT,
null=True)
    #PodKategoria = models.ForeignKey('SubCategory', on_delete=models.PROTECT,
null=True)
    Nazwa_PL = models.CharField(max_length=50)
    Nazwa_EN = models.CharField(max_length=50, default='<Wpisz coś>')
    Nazwa_DE = models.CharField(max_length=50, default='<Wpisz coś>')
    Nazwa_RU = models.CharField(max_length=50, default='<Wpisz coś>')
    Opis_PL = models.CharField(max_length=150)
    Opis_EN = models.CharField(max_length=150, default='<Wpisz coś>')
    Opis_DE = models.CharField(max_length=150, default='<Wpisz coś>')
    Opis_RU = models.CharField(max_length=150, default='<Wpisz coś>')
    Zdjecie = models.ImageField(upload_to='images/')
    Cena = models.CharField(max_length=50)
    Stan = models.BooleanField(default=False)
    Waga = models.CharField(max_length=50)
    Rozmiar = models.CharField(max_length=50)
    Skladniki_PL = models.CharField(max_length=600, default='<Wpisz coś>')
    Skladniki_EN = models.CharField(max_length=600, default='<Wpisz coś>')
    Skladniki_DE = models.CharField(max_length=600, default='<Wpisz coś>')
    Skladniki_RU = models.CharField(max_length=600, default='<Wpisz coś>')
    Dodatki_PL = models.CharField(max_length=600)
    Dodatki_EN = models.CharField(max_length=600, default='<Wpisz coś>')
    Dodatki_DE = models.CharField(max_length=600, default='<Wpisz coś>')
    Dodatki_RU = models.CharField(max_length=600, default='<Wpisz coś>')
    Data = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey(User, verbose_name='Uzytkownik',
on_delete=models.CASCADE)

    def __str__(self):
        return self.Nazwa_PL

class Configuration(models.Model):
    name = models.CharField(max_length=50, db_index = True)
    first_image = models.ImageField(upload_to='images/')
    second_image = models.ImageField(upload_to='images/')
    icon_na_miejscu = models.ImageField(upload_to='images/')
    icon_na_wynos = models.ImageField(upload_to='images/')
```

```

icon_dodatki = models.ImageField(upload_to='images/')
icon_karta = models.ImageField(upload_to='images/')
icon_blik = models.ImageField(upload_to='images/')

def __str__(self):
    return self.name

class Category(models.Model):
    name = models.CharField(max_length=50, db_index = True)
    Kategoria_image = models.ImageField(upload_to='images/', null=True)

    def __str__(self):
        return self.name

class Language(models.Model):
    name = models.CharField(max_length=50, db_index = True)
    Jezyk_image = models.ImageField(upload_to='images/', null=True)

    def __str__(self):
        return self.name

```

## Załącznik 2, Funkcją odpowiadającą za pobieranie dań z serwera

```
def get_meals(Token):
    import requests
    import json

    token = 'Token ' + str(Token)

    #WEB
    r = requests.get('http://3.123.205.81:8000/api/v1/pizza/',
headers={'Authorization': token})

    #LOCAL
    #r = requests.get('http://127.0.0.1:8000/api/v1/pizza/',
headers={'Authorization': token})

    r_dict = json.loads(r.text)

    # print(type(r_dict))
    a = 0
    # print(r_dict)
    print('\n')

    for i in r_dict:
        try:
            print(r_dict[a]['Nazwa_PL'])
            print(r_dict[a]['Zdjecie'])
        except: pass
        print('-----')
    -----')
    a = a+1

    return r_dict
```

### Załącznik 3, Funkcja odpowiadająca za wydruk paragonu

```
def Druknij(self):
    if platform != 'android':
        ser = serial.Serial(WindowManager.COMPORT, 9600, timeout=1.5)

        #Pobieramy czas
        now = datetime.datetime.now()
        year = now.year
        month = now.month
        day = now.day
        hour = now.hour
        minute = now.minute

        #Numer zamówienia
        WindowManager.numer_zamowienia = str(random.randint(1,99))
        self.manager.get_screen("Koncowe").numer_zamowienia =
WindowManager.numer_zamowienia

        # Send the message to the printer
        #\x1d\x21\x06

        ser.write('\x1d\x21\x22\x1b\x61\x01\n\nKusTech'.encode())
        ser.write('\x1d\x21\x11\n\n*****'.encode())
        ser.write('\x1d\x21\x04\nKaravaev Konstantin'.encode())
        ser.write('\nnr.albumu: 303144'.encode())
        ser.write('\nNIP: 521-40-15-102'.encode())
        ser.write('\nPraca inzynierska PW WMT'.encode())
        ser.write('\x1d\x21\x11\n\n*****'.encode())

        ser.write(f'\x1b\x21\x01\x1b\x21\x03\n{year}-{month}-
{day} {hour}-{minute}'.encode())
        ser.write('\x1d\x21\x06\nParagon fiskalny\n\n'.encode())
        ser.write('\x1b\x21\x01'.encode())
        ser.write('\x1b\x21\x03'.encode())

        print(WindowManager.koszyk_list)
        # #Drukujemy menu
        try:
            a = WindowManager.koszyk_list[0][0]
            for el in WindowManager.koszyk_list:
                ser.write(f'{el[0]}\n'.encode())
                ser.write(f'{el[4]}*{el[1].split()[0]}'.encode())
                ser.write(f'\n{int(el[4])*int(el[1].split()[0])}.00 Zl
\n\n'.encode())
            except:
                for el in WindowManager.koszyk_list:
                    ser.write(f'{el[0]} {el[4]}*{el[1]}
{int(el[4])*int(el[1].split()[0])} \n'.encode())
```

```

        #Podsumowanie
        ser.write('\x1d\x21\x11\n#####'.encode())
        ser.write(f'\x1d\x21\x06\n---SUMA PLN:
{WindowManager.cena_koncowa}----'.encode())
        ser.write('\x1d\x21\x11\n#####\n\n'.encode())
        ser.write('\x1d\x21\x06Numer twojego zamowienia:'.encode())
        ser.write(f'\x1d\x21\x22\n\n{WindowManager.numer_zamowienia}'.enco
de())

        ser.write('\x1d\x21\x11\n\n#####'.encode())
        ser.write('\x1d\x21\x06\n\nDziekuje za zakup!\n\n\n\n'.encode())

        # Wait for the printer to finish printing
        response = ser.readline()
        while response:
            print(response.strip().decode())
            response = ser.readline()

        # Close the serial connection
        ser.close()

        #Usuwamy zawartość koszyka
        self.ekran_koszyka = self.manager.get_screen("Koszyk")
        self.ekran_koszyka.ids.koszyk.clear_widgets()
        self.ekran_koszyka.cena_koncowa_koszyk = '0 Zł'
        self.manager.get_screen("WyborPlatnosci").doZaplaty = f'Do zapłaty
0 Zł'

        self.manager.get_screen("Menu_PL").cena_koncowa = f'0 Zł'

        #Dla zamowienia
        WindowManager.idszka = ''
        WindowManager.koszyk_list = []
        WindowManager.ilosc = 1
        WindowManager.cena_z_dodatkami = '0 Zł'
        WindowManager.cena_koncowa = ''
        WindowManager.nazwa_produkту = ''
        WindowManager.rozmiar_dania = ''
        WindowManager.opis = ''
        WindowManager.adres_zdjecia = ''
        WindowManager.skladniki_list = []
        WindowManager.dodatki_dict = {}
    else: pass
    pass

```