|  |  |  |
| --- | --- | --- |
| Dokumentacja projektu | | **AI1** |
| **Autor** | Eliza Tworkowska (119003), Oskar Paśko (117987) | **Ocena** |
| **Kierunek, rok** | Informatyka i Ekonometria, II rok, st. stacjonarne (3,5-l) |  |
| **Temat projektu** | Serwis naprawy telefonów |

Spis Treści

[Wstęp 2](#_Toc138760202)

[Narzędzia i technologie 2](#_Toc138760203)

[Język programowania 3](#_Toc138760204)

[Framework 3](#_Toc138760205)

[Baza Danych 3](#_Toc138760206)

[Front-end 3](#_Toc138760207)

[Środowisko programistyczne 3](#_Toc138760208)

[Baza Danych 3](#_Toc138760209)

[Schemat ERD bazy danych 3](#_Toc138760210)

[Opis bazy danych 4](#_Toc138760211)

[1. Tabela "record\_client": 4](#_Toc138760212)

[2. Tabela "record\_record": 4](#_Toc138760213)

[3. Tabela "record\_service": 4](#_Toc138760214)

[4. Tabela "record\_location": 4](#_Toc138760215)

[5. Tabela "record\_posn": 5](#_Toc138760216)

[6. Tabela "record\_employee": 5](#_Toc138760217)

[7. Tabela "record\_client\_user\_permissions": 5](#_Toc138760218)

[8. Tabela "auth\_permission": 5](#_Toc138760219)

[9. Tabela "auth\_group\_permissions": 5](#_Toc138760220)

[10. Tabela "auth\_group": 6](#_Toc138760221)

[11. Tabela "record\_client\_groups": 6](#_Toc138760222)

[12. Tabela "django\_content\_type": 6](#_Toc138760223)

[13. Tabela "django\_admin\_log": 6](#_Toc138760224)

[Relacje w modelach 7](#_Toc138760225)

[Migracje 7](#_Toc138760226)

[Projekt GUI 8](#_Toc138760227)

[Uruchomienie aplikacji 9](#_Toc138760228)

[Funckjonalności aplikacji 10](#_Toc138760229)

[Rejestracja 10](#_Toc138760230)

# Wstęp

Projekt serwisu naprawy telefonów oparty jest na frameworku Django, który zapewnia wydajność, skalowalność i bezpieczeństwo. Serwis ma na celu umożliwienie klientom łatwego dostępu do informacji o firmie oraz cenach napraw, a także zapewnia możliwość tworzenia nowych kont klientów, logowania się, monitorowania statusu napraw oraz składania nowych zleceń. Ponadto, klienci mają możliwość zarządzania swoimi danymi osobowymi.

Opis funkcjonalności:

1. Główna strona:
   * Wyświetla podstawowe informacje o firmie, takie jak historia, misja i wartości.
   * Zapewnia nawigację do innych sekcji serwisu.
2. Strona z cenami napraw:
   * Wyświetla pełną listę dostępnych usług naprawy wraz z ich cenami.
   * Umożliwia klientom łatwe porównanie cen i wybór odpowiedniej usługi.
3. Rejestracja i logowanie:
   * Klienci mogą utworzyć nowe konto poprzez wypełnienie formularza rejestracyjnego.
   * Istniejący klienci mogą zalogować się, używając swojego adresu e-mail i hasła.
4. Panel klienta:
   * Zalogowani klienci mają dostęp do swojego panelu klienta, gdzie mogą zarządzać swoim kontem.
   * Pozwala klientom na podgląd statusu swoich napraw, takich jak "w trakcie naprawy" lub "gotowe do odbioru".
5. Zarządzanie danymi osobowymi:
   * Klienci mają możliwość edycji swoich danych osobowych, takich jak adres, numer telefonu itp.
   * Pozwala klientom zmieniać hasło dostępu do swojego konta.

# Narzędzia i technologie

Projekt wykorzystuje następujące technologie i narzędzia:

## Język programowania

* Python (3.11.3): język programowania wysokiego poziomu ogólnego przeznaczenia. Strona <https://www.python.org/>

## Framework

* Django (4.0.2): darmowy i otwarty framework przeznaczony do tworzenia aplikacji internetowych, napisany w Pythonie. ([https://www.djangoproject.com](https://www.djangoproject.com/)/)

## Baza Danych

* SQLite: otwartoźródłowy system zarządzania relacyjną bazą danych oraz biblioteka C implementująca taki system, obsługująca SQL. (<https://www.sqlite.org/index.html>)

## Front-end

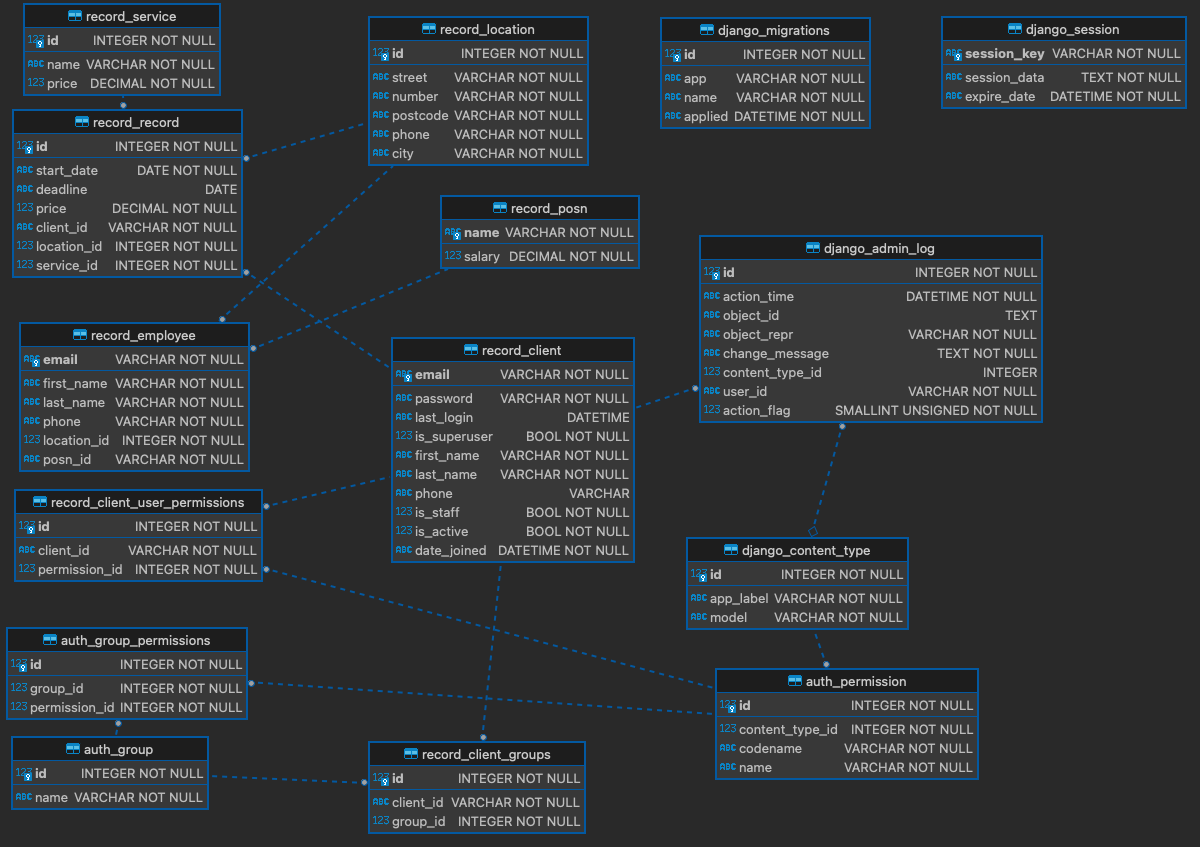
* Bootstrap: To framework CSS ułatwiający tworzenie responsywnego i atrakcyjnego interfejsu. Najnowsza wersja to 5.3 (<https://getbootstrap.com/>)

## Środowisko programistyczne

* Visual Studio Code – darmowy edytor kodu źródłowego z kolorowaniem składni dla wielu języków, stworzony przez Microsoft, o otwartym kodzie źródłowym

# Baza Danych

## Schemat ERD bazy danych



## Opis bazy danych

Baza danych jest połączeniem bazy danych aplikacji i domyślnej bazy danych Django, generowaną automatycznie przy starcie projektu.

## 1. Tabela "record\_client":

Przechowuje informacje o klientach serwisu, takie jak adres e-mail, hasło, dane osobowe, numer telefonu i datę rejestracji.

* email (klucz główny, varchar not null)
* password (varchar not null)
* last\_login (Datetime)
* is\_superuser (bool not null)
* first\_name (varchar not null)
* last\_name (varchar not null)
* phone (varchar)
* is\_staff (bool not null)
* is\_active (bool not null)
* date\_joined (datetime not null)

## 2. Tabela "record\_record":

Zawiera informacje o poszczególnych naprawach telefonów, takie jak data rozpoczęcia, termin wykonania, koszt, identyfikator klienta, lokalizację naprawy i usługę, którą trzeba wykonać. Klucze obce łączą tę tabelę z tabelami "record\_client", "record\_location" i "record\_service".

* id (integer not null)
* start\_date (date not null)
* deadline (date)
* price (decimal not null)
* client\_id (varchar not null, klucz obcy do "record\_client")
* location\_id (integer not null, klucz obcy do "record\_location")
* service\_id (integer not null, klucz obcy do "record\_service")

## 3. Tabela "record\_service":

Przechowuje informacje o usługach naprawy dostępnych w serwisie, takie jak nazwa usługi i jej cena.

* id (integer not null)
* name (varchar not null)
* price (decimal not null)

## 4. Tabela "record\_location":

Zawiera dane o lokalizacjach serwisu naprawy telefonów, takie jak ulica, numer, kod pocztowy, numer telefonu i miasto.

* id (integer not null)
* street (varchar not null)
* number (varchar not null)
* postcode (varchar not null)
* phone (varchar not null)
* city (varchar not null)

## 5. Tabela "record\_posn":

Przechowuje informacje o stanowiskach w firmie, takie jak nazwa stanowiska i wynagrodzenie.

* name (varchar not null, klucz główny)
* salary (Decimal not null)

## 6. Tabela "record\_employee":

Zawiera dane pracowników serwisu, takie jak adres e-mail, dane osobowe, numer telefonu, lokalizację w której pracują, i stanowisko, które zajmują. Klucze obce łączą tę tabelę z tabelami "record\_location" i "record\_posn".

* email (varchar not null, klucz główny)
* first\_name (varchar not null)
* last\_name (varchar not null)
* phone (varchar not null)
* location\_id (integer not null, klucz obcy do "record\_location")
* posn\_id (varchar not null, klucz obcy do "record\_posn")

## 7. Tabela "record\_client\_user\_permissions":

Przechowuje informacje o uprawnieniach użytkowników klientów, takie jak identyfikator klienta i identyfikator uprawnienia. Klucze obce łączą tę tabelę z tabelami "record\_client" i "auth\_permission".

* id (integer not null, klucz główny)
* client\_id (varchar not null, klucz obcy do "record\_client")
* permission\_id (integer not null, klucz obcy do "auth\_permission")

## 8. Tabela "auth\_permission":

Zawiera informacje o uprawnieniach w systemie uwierzytelniania Django, takie jak identyfikator typu zawartości, kod, nazwa i inne szczegóły.

* id (integer not null, klucz główny)
* content\_type\_id (integer not null)
* codename (varchar not null)
* name (varchar not null)

## 9. Tabela "auth\_group\_permissions":

Przechowuje informacje o przypisanych uprawnieniach do grup w systemie uwierzytelniania Django. Łączy klucze obce z tabelami "auth\_group" i "auth\_permission".

* id (integer not null, klucz główny)
* group\_id (integer not null)
* permission\_id (integer not null)

## 10. Tabela "auth\_group":

Zawiera informacje o grupach użytkowników w systemie uwierzytelniania Django, takie jak nazwa grupy.

* id (integer not null, klucz główny)
* name (varchar not null)

## 11. Tabela "record\_client\_groups":

Przechowuje informacje o przypisanych grupach klientów w serwisie naprawy telefonów. Łączy klucze obce z tabelami "auth\_group" i "record\_client".

* id (integer not null, klucz główny)
* client\_id (varchar not null, klucz obcy do "record\_client")
* group\_id (integer not null, klucz obcy do "auth\_group")

## 12. Tabela "django\_content\_type":

Przechowuje informacje o typach zawartości w Django, takie jak etykieta aplikacji i nazwa modelu.

* id (integer not null, klucz główny)
* app\_label (varchar not null)
* model (varchar not null)

## 13. Tabela "django\_admin\_log":

Zawiera logi działań w panelu administracyjnym Django, takie jak czas akcji, identyfikator obiektu, nazwa obiektu,

* id (integer not null, klucz główny)
* action\_time (datetime not null)
* object\_id (text)
* object\_repr (varchar not null)
* change\_message (text not null)
* content\_type\_id (integer)
* user\_id (varchar not null)
* action\_flag (smallint unsigned not null)

## Relacje w modelach

W naszej bazie danych występują następujące główne relacje:

1. Tabela "record\_record" ma relacje typu One-To-Many z tabelami "record\_client", "record\_location" i "record\_service". Oznacza to, że wiele rekordów napraw może być powiązanych z jednym klientem, jedną lokalizacją i jedną usługą naprawy.
2. Tabela "record\_employee" ma relację typu One-To-Many z tabelą "record\_location". Oznacza to, że wiele pracowników może być przypisanych do jednej lokalizacji.
3. Tabela "record\_employee" ma również relację typu One-To-Many z tabelą "record\_posn". Oznacza to, że wiele pracowników może mieć to samo stanowisko.
4. Tabela "record\_client\_user\_permissions" ma relację typu One-To-Many z tabelą "record\_client" i "auth\_permission". Oznacza to, że jeden klient może mieć wiele przypisanych uprawnień.
5. Tabela "auth\_group\_permissions" ma relację typu One-To-Many z tabelami "auth\_group" i "auth\_permission". Oznacza to, że jedna grupa może mieć wiele przypisanych uprawnień.
6. Tabela "record\_client\_groups" ma relację typu One-To-Many z tabelami "auth\_group" i "record\_client". Oznacza to, że jeden klient może być przypisany do wielu grup, a jedna grupa może mieć wielu klientów.

Wszystkie te relacje są typu One-To-Many, co oznacza, że wiele rekordów z jednej strony relacji może być powiązanych z jednym rekordem z drugiej strony relacji.

## Migracje

Django generuje automatycznie migracje po wpisaniu komendy „py manage.py makemigrations” z zamodelowanej bazy w pliku model.py.

# Projekt GUI

A screenshot of a phone service

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Uruchomienie aplikacji

Aby uruchomić aplikację należy wykonać następujące czynności:

1. Komputer musi mieć zainstalowany język Python i framework Django, menadżer pakietów PIP. By zainstalować Pythona ściągamy go z strony producenta, natomiast PIP powinien być zainstalowany domyślnie z Pythonem. Django następnie instalujemy poprzez komendę ‘pip install django’ w wierszu poleceń.
2. Upewniamy się czy Django jest zainstalowany poprzez komendę django-admin w wierszu poleceń
3. Pobierz oraz rozpakuj projekt
4. Otwieramy wiersz poleceń w folderze tego projektu tak żeby był tam plik manage.py
5. W wierszu poleceń wpisujemy „py manage.py runserver” co uruchamia serwer na adresie <http:///127.0.0.1:8000/>

Po wykonaniu tych wszystkich kroków będzie można swobodnie poruszać się i korzystać ze strony internetowej.

# Funckjonalności aplikacji

## Rejestracja

A screenshot of a login form

Description automatically generated with medium confidence

Nowy klient może się zarejestrować.

A screenshot of a login form

Description automatically generated with medium confidence

Próba kliknięcia przycisku rejestruj bez wpisania wymaganych danych spowoduje wyświetlenie się podpowiedzi.

Po udanej rejestracji zostajemy przekierowani na stronę główną.

## Logowanie

A screen shot of a phone

Description automatically generated with low confidence

Przykładowe konta do logowania:

* test@test.com / TrudneHaslo
* jkowalski@cos.com / SlabeHaslo
* anowak@cos.com / WlasneHaslo

## CRUD Administrator

A screenshot of a computer

Description automatically generated

Dodawanie Pracownika

A screenshot of a computer

Description automatically generated

Edytowanie pracownika

A screenshot of a computer

Description automatically generated

Usuwanie pracownika

## Zarządzanie użytkownikami

A screenshot of a computer

Description automatically generated