Implementacja systemu i wykonanie testów sprawdzających

Etap IV | Grupa nr.4

|  |  |
| --- | --- |
| **Produkt etapu** | **Autor** |
| Implemetacja bazy danych | Tomasz Jarnutowski |
| Implementacja warstwy logicznej |  |
| Implementacja GUI |  |
| Gotowy system informatyczny |  |
| Wyniki przeprowadzonych testów |  |

Spis treści

[1. Implementacja bazy danych 3](#_Toc121337627)

[1.1. Diagram bazy danych 3](#_Toc121337628)

[1.2. Sposób implementacji bazy danych 4](#_Toc121337629)

[2. Implementacja warstwy logicznej 5](#_Toc121337630)

[3. Implementacja GUI 5](#_Toc121337631)

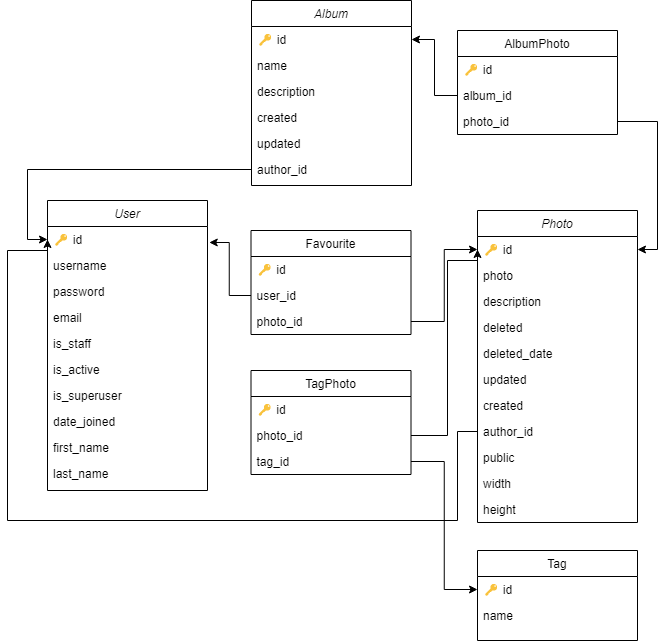
[4. Gotowy system informatyczny 5](#_Toc121337632)

[5. Wyniki przeprowadzonych testów 5](#_Toc121337633)

# 

# Implementacja bazy danych

## Diagram bazy danych



## Sposób implementacji bazy danych



Baza została zaimplementowana z użyciem DjangoORM. Tabele są reprezentowana za pomocą klas, a zmienne odpowiadają nazw kolumn. Funkcje „\_\_str\_\_” pozwalają na łatwiejszy odczyt wpisów w przypadku użycia domyślnego panelu administracyjnego. DjangoORM domyślnie sam dodaje kolumnę „ID”, dlatego nie ich w implementacji powyżej. Na powyższym podglądzie nie ma również tabeli User, jest to spowodowane faktem że w projekcie jest użyty domyślny model użytkownika który jest dostarczany przez framework.

Tabele określające relacje wiele do wielu są tworzone z poziomu zmiennej klasy(models.ManyToManyField) i nie ma potrzeby tworzenia dodatkowej klasy. Relacja typu jeden do wielu jest reprezentowaną za pomocą models.ForeignKey. W przypadku pól typu dat, używam opcji „auto\_now” – która aktualizację datę przy każdej edycji wpisu oraz „auto\_now\_add” która dodaję datę w momencie tworzenia wpisu.

# Implementacja warstwy logicznej

# Implementacja GUI

# Gotowy system informatyczny

# Wyniki przeprowadzonych testów