# Sprawozdanie

## Standardy kodowania:

- Jasne nazewnictwo zmiennych i funkcji.
- Puste linie: Używanie dwóch pustych linii przed definicjami klas i jednej przed definicjami funkcji wewnątrz klas aby poprawić czytelność.
- Stosowanie odpowiednich odstępów wokół operatorów, po przecinkach i dwukropkach.
- Definicje klas z dużej litery.
- Nazwy klas, funkcji, danych tekstowych po angielsku.

### Standardy dokumentowania:

W kodzie nie używano komentarzy, ale nazwy klas i funkcji opisują ich działanie.

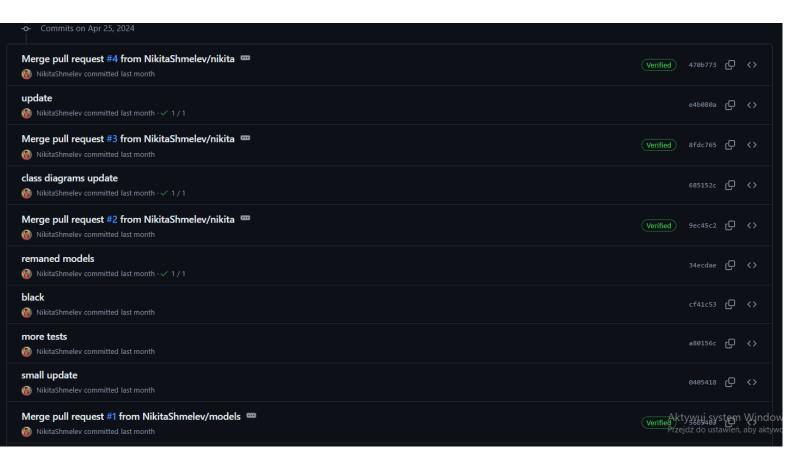
#### Język implementacji i IDE:

Język programowania: **Python**. Wybrany ze względu na swoją prostotę, czytelność, dużą ilość bibliotek oraz zaznajomienie z nim. Użyto framework **unittest**, moduły **re** i **datetime.** 

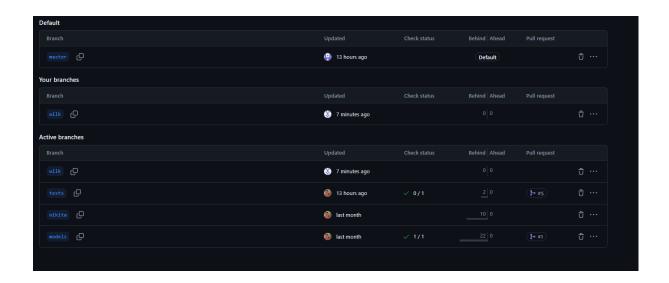
IDE: PyCharm

## Wykorzystanie repozytorium:

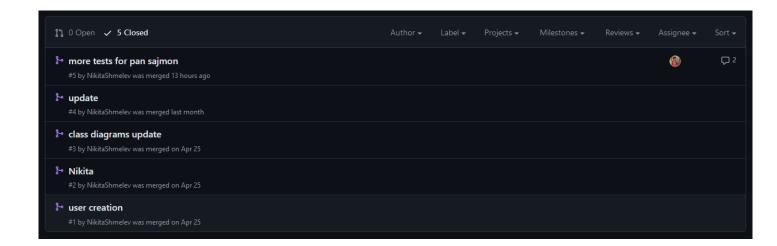
Założono repozytorium w witrynie Github. Skorzystano z wielu funkcji Githuba, m.in. takich jak pull request, branch.



Przykładowe wykonane commity. W committach znajdowały się zmiany w kodzie, nowe testy oraz aktualizacje diagramów.



Utworzone branche. W branch "tests" napisano testy jednostkowe które zostały potem scalone do głównego branchu.



Wykonane pull requesty.

#### Testy jednostkowe:

Wykonano 27 testów jednostkowych, użyto framework testowania jednostkowego **unittest** Obsługuje automatyzację testów i współdzielenie kodu przygotowującego i kończącego testy.

#### Testy weryfikują:

- Czy użytkownik jest poprawnie rejestrowany.
- adres email.
- proces zmiany hasła
- proces rejestracji administratora i
- rejestrację kupującego i sprzedawcy
- poprawność tworzenia przedmiotu.
- generowanie unikalnych identyfikatorów dla przedmiotów.
- możliwość edytowania szczegółów przedmiotu.
- proces usuwania przedmiotu.
- proces zakupu przedmiotu przez kupującego.
- poprawność tworzenia transakcji.
- proces rejestracji transakcji, zakończenia transakcji i zmiany statusu przedmiotu.

#### TDD:

Testy były pisane już od początku otwarcia repozytorium, wraz z resztą kodu, np. w committach "fields update" i "more tests".

```
35 + user = User(
36 + name="Nikita",
37 + surname="Shmelyov",
38 + phone_number=123,
39 + password="password"
40 + )
41 +
```

```
def test_user_change_password(self):
64
               email = "new1@gmail.com"
               user = User(**{**self.base_data, "email": email})
65
66
               user.register()
               self.user_was_created(user=user)
67
68
               user.change_password(
70
                   old_password=self.base_data["password"], new_password="123"
72
               with self.assertRaises(PasswordValidationException):
                   user.change_password(
74
75
                       old_password=self.base_data["password"], new_password="123"
77
78
79
    + class TestAdmin(TestUser):
           def test_admin_creation(self):
81
               admin = Admin(**self.base_data)
82
               admin.register()
83
               self.user_was_created(user=admin)
84 +
               self.assertEqual(admin.admin, True)
85
86
87
    + class TestBuyer(TestUser):
89
           def test_buyer_creation(self):
               buyer = Buyer(**self.base_data)
90
               buyer.register()
               self.user_was_created(user=buyer)
93
    + class TestSeller(TestUser):
95
           def test_seller_creation(self):
96
               seller = Seller(**self.base_data)
97
               seller.register()
               self.user_was_created(user=seller)
```