

**Ministerul Educației și Cercetării al Republicii
Moldova Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și
Microelectronică**

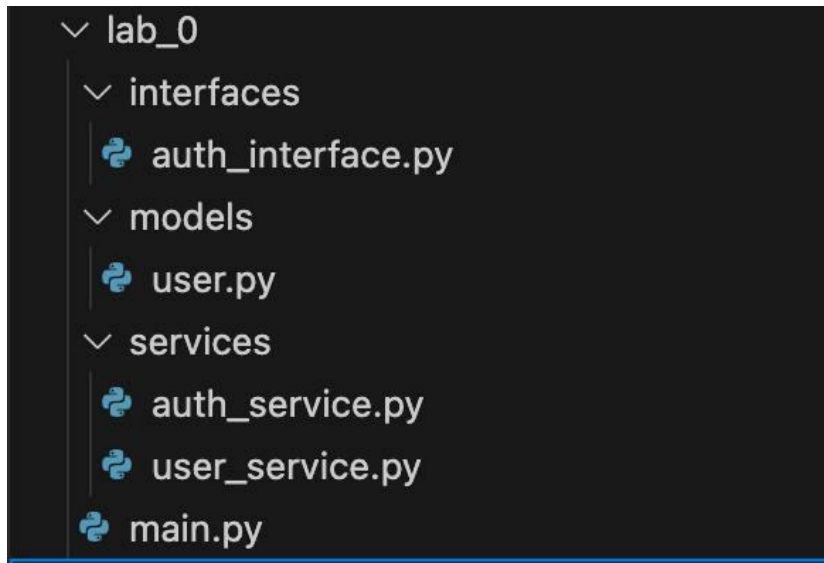
**Tehnici si mecanisme de
proiectare software
Laborator work 0**

**Elaborated:
st. gr. FAF-222**

Dimbitchi Sergiu

Scopul lucrări :

Exercitiile practice :



Explicația exercițiului :

```
auth_interface.py ×
lab_0 > interfaces > auth_interface.py > ...
1  # interfaces/auth_interface.py
2  from abc import ABC, abstractmethod
3
4
5  class AuthInterface(ABC):
6      @abstractmethod
7      def login(self, username: str, password: str) -> bool:
8          pass
9
10     @abstractmethod
11     def logout(self) -> None:
12         pass
```

auth_interfaces.py - defines an abstract base class called `AuthInterface` using Python's `abc` module. It specifies two abstract methods: `login`, which takes a username and password as parameters and returns a boolean indicating success or failure, and `logout`, which does not take any parameters and does not return a value. Classes that inherit from `AuthInterface` must implement these methods to provide concrete authentication functionality.

```

user.py x
lab_0 > models > user.py > ...
1 # models/user.py
2 class User:
3     def __init__(self, username: str, password: str, email: str):
4         self.username = username
5         self.password = password
6         self.email = email
7

```

user.py - defines a `User` class that represents a user in an application. The `__init__` method initializes a new `User` object with three attributes: `username`, `password`, and `email`, which are all expected to be of type `str`. When a new instance of the `User` class is created, these attributes are set to the values passed as arguments to the constructor.

```

auth_service.py x
lab_0 > services > auth_service.py > ...
1 # services/auth_service.py
2 from interfaces.auth_interface import AuthInterface
3
4
5 class AuthService(AuthInterface):
6     def __init__(self):
7         self.logged_in = False
8
9     def login(self, username: str, password: str) -> bool:
10        # Example authentication logic
11        if username == "user" and password == "password":
12            self.logged_in = True
13            print("User authenticated!")
14            return True
15        print("Authentication failed.")
16        return False
17
18    def logout(self) -> None:
19        self.logged_in = False
20        print("User logged out.")

```

auth-service.py - defines a class `AuthService` that inherits from the `AuthInterface` abstract base class. In the `__init__` method, it initializes an instance variable `logged_in` to `False`, indicating that the user is not currently logged in.

- The `login` method takes a username and password, checks them against the values "user" and "password", and, if they match, sets `logged_in` to `True`, prints a success message, and returns `True`. If authentication fails, it prints a failure message and returns `False`.

- The `logout` method sets `logged_in` to `False` and prints a message indicating the user has logged out.

```
user_service.py X
lab_0 > services > user_service.py > ...
1  # services/user_service.py
2  from models.user import User
3
4
5  class UserService:
6      def __init__(self):
7          self.users = []
8
9      def add_user(self, user: User) -> None:
10         self.users.append(user)
11         print("User added successfully!")
12
13     def get_user_by_username(self, username: str) -> User:
14         for user in self.users:
15             if user.username == username:
16                 return user
17         return None
18
```

user_service.py - defines a class `UserService` that manages a list of `User` objects. In the `__init__` method, it initializes an empty list called `users` to store the user instances.

- The `add_user` method takes a `User` object as a parameter, appends it to the `users` list, and prints a success message.

- The `get_user_by_username` method searches through the `users` list for a user with a matching username; if found, it returns that `User` object. If no user is found with the given username, it returns `None`.

```
main.py ×
lab_0 > main.py > ...
1  # main.py
2  from services.user_service import UserService
3  from services.auth_service import AuthService
4  from models.user import User
5
6  # Initialize services
7  user_service = UserService()
8  auth_service = AuthService()
9
10 # Add a user
11 new_user = User("user", "password", "user@example.com")
12 user_service.add_user(new_user)
13
14 # Authenticate user
15 username = "user"
16 password = "password"
17
18 user = user_service.get_user_by_username(username)
19
20 if user:
21     if auth_service.login(user.username, password):
22         print(f"Welcome, {user.username}!")
23     else:
24         print(f"Authentication failed for {user.username}")
25 else:
26     print("User not found.")
27
```

main.py :

Service Initialization: It starts by importing the necessary classes and initializing instances of `UserService` and `AuthService`.

User Creation and Addition: A new `User` object (`new_user`) is created with a username, password, and email. This user is then added to the `UserService` using the `add_user` method.

User Authentication:

- The code defines the `username` and `password` variables for authentication.
- It attempts to retrieve the user from `UserService` by calling `get_user_by_username(username)`. If the user is found, it proceeds to authenticate the user using `auth_service.login()`.
- If authentication is successful, it prints a welcome message. If authentication fails, it prints an error message indicating that authentication failed for the user.
- If the user is not found, it prints "User not found."