The MVC architecture will be used to develop a web application. The application is divided into three main components, each of which is responsible for different tasks.

Veranika Aliashkevich
(Fr. Skorina GSU, Gomel)
Scientific advisor **Viktar Liauchuk**, Ph.D. in technics, associate professor

OVERVIEW OF THE INFORMATION LOGICAL DATA MODEL FOR DANCES STUDIO «RAISKIY»

The input information for the content of the site was provided by the studio's licensing and regulatory documents, regulations on activities, photographs from studio events, price lists of subscriptions, and additional information. Also, the input information is the registration form for a trial lesson for unauthorized users and the login form for the personal account of an already created user.

The output information represents various pages that the user of the developed web application will see.

The first page is the «Main page» of the web application, it contains the main photo presentation of the studio, which is the main advertisement. Scrolling down, you can see photo galleries with photos from the studio's latest annual reporting concerts, and even below is all the information about subscriptions: what they are and their prices.

The next page is a page with directions for dance in the studio. On it there is a list of all possible directions on the left, after selecting a user, detailed information about the selected style appears in the window on the right. In the place «Information about the selected direction» there will be a short description of the style, photos from the lessons of this style, a list of teachers who teach this style, and information about the time of classes. Also below there are comments about the studio and various directions. In the picture with the layout, the dark rectangle indicates the ability of only authorized users: to leave comments. Unauthorized users do not have this form.

The «Trainers» page has the same structure as the page with information about dance directions in the studio, but there is no way to browse and leave comments on it. In place «Information about selected trainer» there will be a short biography of the teacher, photos from classes with this teacher, a list of the styles that he teaches.

The schedule page is a table with the days of the week and times, where the corresponding activity is indicated at the intersection. For unauthorized users to the right of the schedule, there is an opportunity to choose a form for registering a trial lesson in the studio. And for authorized users, instead, there is a button that is responsible for filtering the schedule to get an individual user's schedule, and below this button is information about the number of remaining classes on the active subscription.

The last page has all the information you need to find and contact the studio with ease. It contains the contacts and addresses of the studio, just below the exterior of the studio and its halls, and even below you can find a map to which the location of the studio is tied.

Chaslau Averchanka
(Fr. Skorina GSU, Gomel)
Scientific advisor **Viktar Liauchuk**, Ph.D. in technics, associate professor

TESTING THE WEB APPLICATION FOR A WORKSHOP

There are scenarios of user interaction with the server in this project: from authorization to processing requests. Several scenarios are considered here. Before writing tests, you need to prepare a data area that interacts in some way in tests. For this a backup database was created, and several SQL-scripts were written. They are shown in figures 1 and 2.

```
delete from user_role;
delete from users;

insert into users(id, active, password, username) values
(1, true, '$2a$08$XUSWXBtHO7N4SLkv6VlXWeVguDlQryMS.u9igSmkc30eZyUYYJ25a', 'Cheslav'),
(2, true, '$2a$08$XUSWXBtHO7N4SLkv6VlXWeVguDlQryMS.u9igSmkc30eZyUYYJ25a', 'test');

insert into user_role(user_id, roles) values
(1, 'USER'), (1,'ADMIN'),
(2,'USER')
```

Figure 1 – SQL- script that updates user data

```
@Test
@Sql(value = {"/create-user-before.sql", "/forms-list-before.sql"}, executionPhase = Sql.ExecutionPhase.BEFORE_TEST_METHOD)
@Sql(value = {"/forms-list-after.sql", "/create-user-after.sql"}, executionPhase = Sql.ExecutionPhase.AFTER_TEST_METHOD)
public void correctLoginTest() throws Exception{
    this.mockMvc.perform(formLogin().user("Cheslav").password("test"))
            .andDo(print())
            .andExpect(status().is3xxRedirection())
            .andExpect(redirectedUrl( expectedUrl: "/"));
}
```

Figure 2 – Test that checks user data for validity for authorization