1222·2022 800 ANNI

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Web Applications A.Y. 2023-2024
Homework 1 – Server-side Design and Development

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: 29 April, 2024

Group Acronym	WEBEDGE	
Last Name	**First Name**	**Badge Number**
CAKMAKCI	UMUT BERK	2071408
YANOGLU	MELTEM	2071545
BIMAJ	KEJSI	2112145
TIKHONOV	VLADISLAV	2106920
MUHAMMAD	ALI	2071499

1 Objectives

The ongoing project, extending from the Foundation of Databases course, aims to construct a comprehensive web interface for efficiently managing all aspects of a movie application, including user interactions and member relationships. The developed website enables users to personalize their interface, add preferred movies, curate a watchlist, access information about actors and directors, and explore a catalog of popular films.

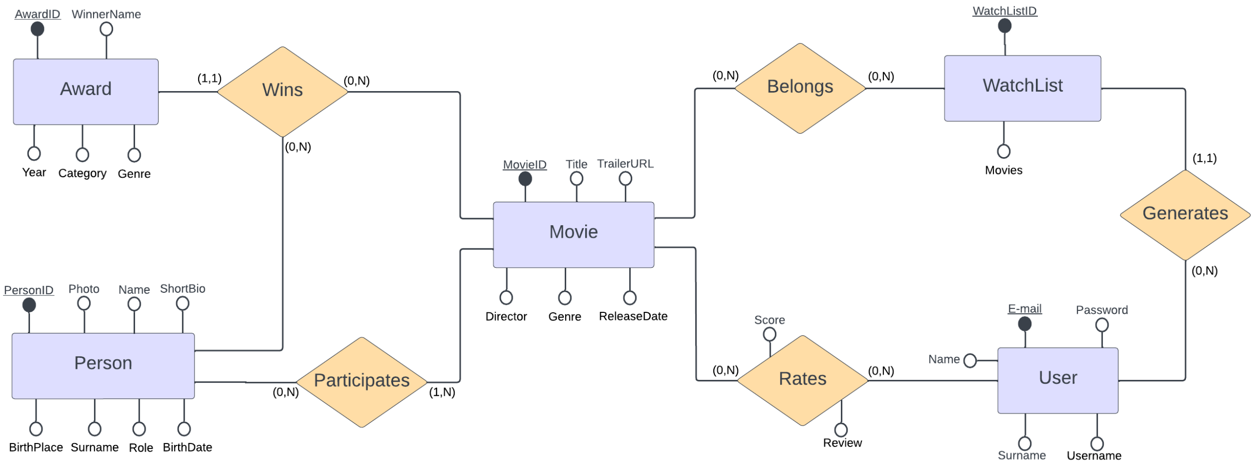
2 Main Functionalities

The web application is tailored to enhance user engagement with the movie catalog, offering a seamless experience for exploring, rating, and interacting with movies. Users can delve into the extensive movie collection, rate films, share their opinions through comments, curate personalized lists of favorite movies, and access detailed information about the cast and crew, thereby eliminating the need to visit multiple sources or cinemas. The website is structured into four key areas:

- **Public Area:** Accessible to all users, the homepage serves as a gateway to the movie collection, showcasing top-rated films and providing an overview of the available titles.
- **User Area:** Users have access to a range of functionalities, including the ability to:
  - Can register the website if they are not registered.
  - Update their personal information.
  - Explore movies through various options, such as watchlists and awards, and access general movie details.
  - Browse and contribute movie reviews, sharing their insights and opinions.
  - Manage their watchlists by adding or removing movies based on their preferences.
- **Movie Area:** This section offers a comprehensive repository of movie-related information, including:
  - Detailed movie profiles featuring essential details like title, director, and cast.
  - Opportunities for users to read existing reviews and submit their own, fostering a community-driven discussion around films.
  - Convenient options for users to add movies to their watchlists for future reference and viewing.
- **Person Area:** Dedicated to providing insights into the individuals behind the scenes, this area offers:
  - Detailed profiles of actors/actresses, directors, writers and producers, showcasing their career highlights and contributions to the film industry.
  - Information about awards they have received for their work across various films, highlighting their achievements and recognition.
  - A curated list of films in which they have played a significant role, offering users a deeper understanding of their filmography and artistic journey.
- **Admin Area:** The admin area is a web interface where the application owner manages movies, actresses, actors, directors, and related content. It allows adding, updating, and removing information about actresses, movie descriptions, actor details, and director profiles.

3 Data Logic Layer

3.1 Entity-Relationship Schema



The ER schema contains 5 Entities;

- **Movie:** Represents the information about the movies. The primary key is the movie identifier (movieID), which is an INTEGER field. The other attributes are title (TEXT), trailer URL (TEXT), releaseDate (DATE), genre (TEXT), and director (TEXT). All attributes here cannot be null. Each movie should have at least one person, i.e. each movie has a director. So, a movie is participated by 1-N person (with relationship *Participates*). A movie also can win 0-N awards (with relationship *Wins*), belongs to 0-N watchlist (with relationship *Belongs*), and rated by 0-N user (with relationship *Rates*).

- **Person:** Represents all actors/actresses/directors/writers/producers that are part of a movie. The primary key is the person identifier (PersonID), which is an INTEGER field. Other attributes include personal information such as name, surname, short biography, birthplace, all having TEXT fields, BirthDate which is a DATE field, and Photo contains a link to the person's photo. The attribute photo can be null, while others cannot. Each person participates 0-N movie (with relationship *Participates*), and can win 0-N awards (with relationship *Wins*).

- **User:** Represents all users that are registered on the movie website. The primary key is email, which is a TEXT field. Other attributes include personal information such as name, surname, password, and username, all having TEXT fields. All attributes here cannot be null. Each user can rate 0-N movies (with relationship *Rates*) and generates 0-N watchlists (with relationship *Generates*).

- **Award:** Represents all awards that an actress/director has gained. The primary key is Award Identifier (AwardID), which is an INTEGER field. Other attributes are WinnerName, Category, Genre all having TEXT fields, and Year, which is a DATE field. Each award is in a 1-1 relationship with either Movie or Person with turnery relationship *Wins*. As a consequence, Award contains as foreign keys both the movie ID and person ID. If award is won by a movie, then person ID field can be null. If award is won by a person, that person needs to be part of a movie and hence movie ID cannot be null.

- **WatchList:** Represents movies waiting to be watched. The primary key is WatchList Identifier (WatchListID), which is an INTEGER field. Another attribute is movies, which is an array list field. Each watchlist is in a 1-1 relationship with User with binary relationship *Generates*. As a consequence, Watchlist contains as a foreign key of user email and it cannot be null. Each watchlist contains 0-N movies (with relationship *Belongs*).

4 Presentation Logic Layer

The following pages are an example of the pages available:

- **Home Page:** This serves as the landing page for the website. It showcases top-rated movies and provides a link to the login page for users. The page features a menu bar accessible across the site post-login, offering options such as Movies/TV shows, Celebrities, and Awards and Events. Additionally, it includes a search bar allowing users to specify their search by title, episode, celebrities or companies.

- **Movie Page:** This page displays all the information about the movie, including the title, actors/actresses, director, reviews, rating and the link where we can watch the trailer of the movie.

- **Login Page:** Users are required to enter their email address, name, surname and password to access the web application's features and content.

- **Person Page:** This section allows users to access information about actors/actresses, directors, writers and the movies they are associated with. Users can view details such as filmography, awards won, and biography.

- **Add/Delete Movies (Admin Page):** This page allows the admin to add or delete movies to the database.

- **Add Persons (Admin Page):** This page allows the admin to add persons (actors/actresses, directors, writers) to the database. Admin can create person form such as name, surname, bio, role, birth place, birth date, photo link.

- **Watchlists (User Page):** This feature enables users to curate a list of movies they intend to watch later. It serves as a convenient reminder for users to keep track of movies of interest.

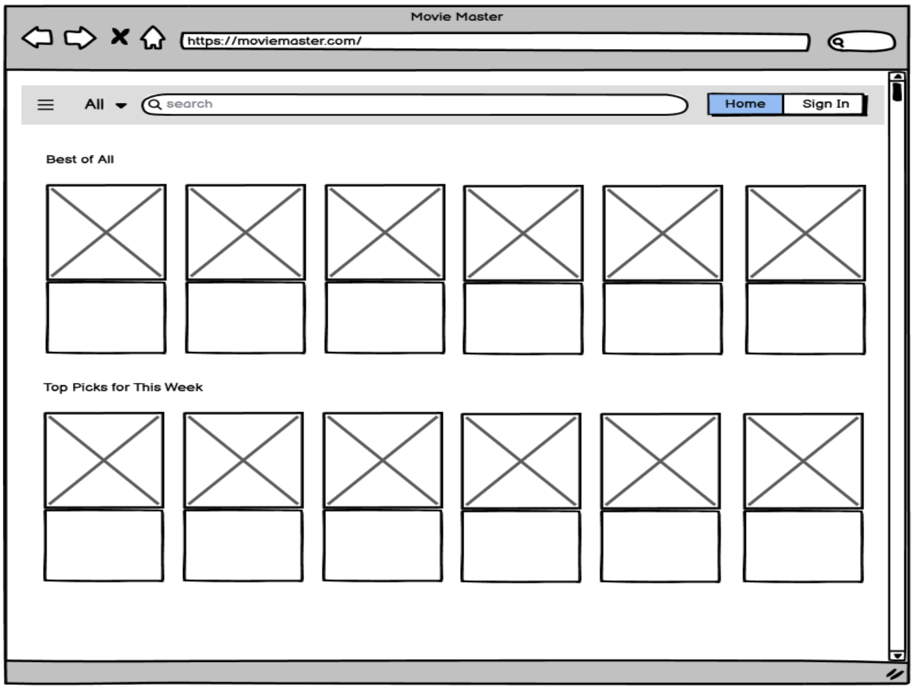
- **Your Ratings (User Page):** Users can view statistics related to movie ratings, such as the average rating of movies viewed by the user.

- **User Profile (User Page):** This page allows users to view and modify their personal information stored on the platform. Users can update details such as their first name, last name, username, email address and password.

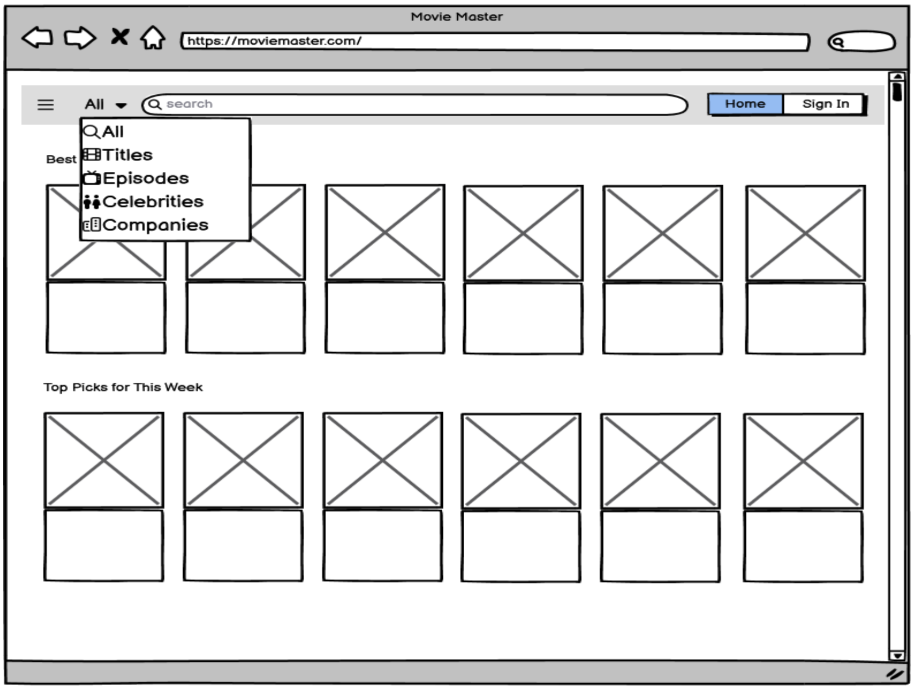
- **Your Lists (User Page):** The user of the application can add movies to his lists of movies, create a new list or delete them.

4.1 Home Page

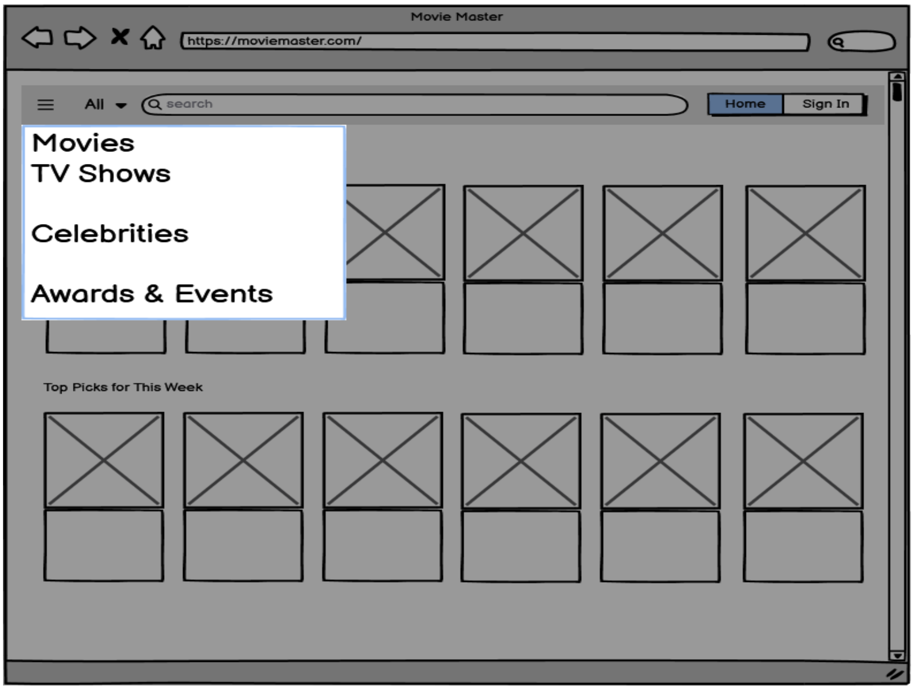
The home page is the main page of our website. In this page, we have some movie suggestions for 2 categories; best of all and top picks for this week. We choose best of all movies based on overall ratings and display the first 6 of them in our home page. For top picks, admins choose randomly (or last week's most watched first 6 movies) and they are also displayed in home page.



Home page contains a search bar to search movies/tv shows/celebrities directly. They can search from all of our content or they can narrow down the search area by selecting the specific category. They can select the category via submenu leftside of the search bar.

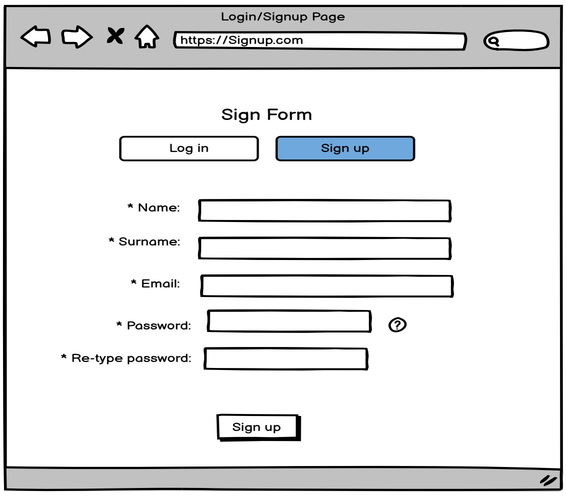


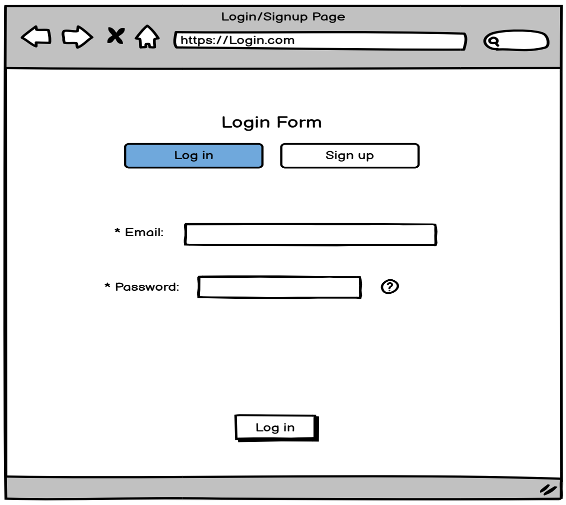
Also, if the user wants to go to the related main page (movies, tv shows, celebrities, etc.), they simply can click the 3-line submenu button and select the category they want.



We also have 2 other buttons on the Home page. One of them is Home button and the other is Sign In button. Sign In button directs user to Login/Sign Up pages, and user can choose which one is suitable for him/her. Home page directs users to home page regardless of which page they are.

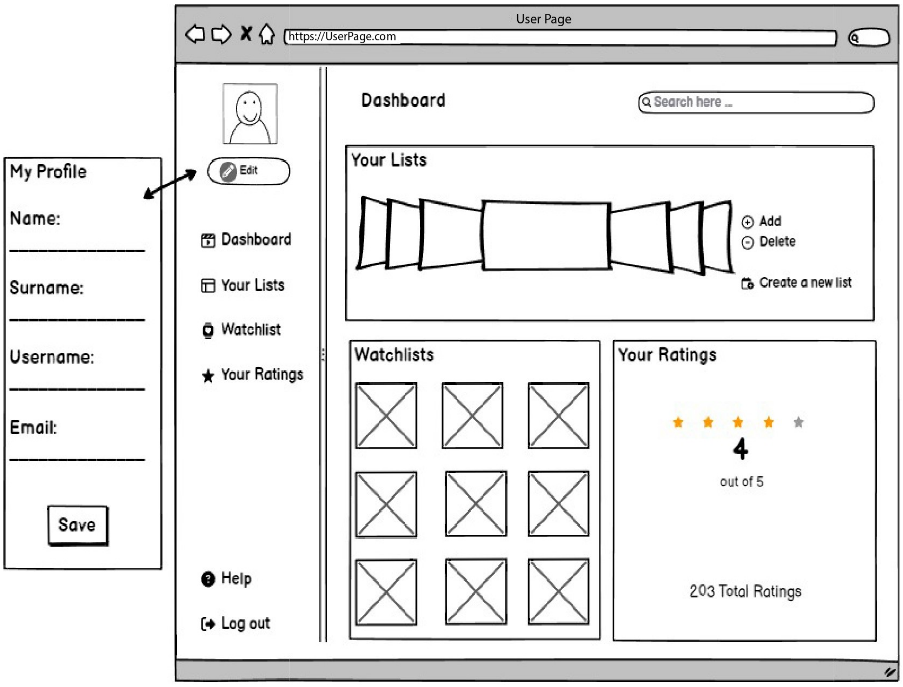
4.2 Login/Signup Page





The login page is the gateway to authenticate users by requiring them to input their credentials —an email address and a password— before granting access to the platform's features and content. The form consists of two boxes, one to gather the username and the other for the password. The signup page serves as the entry point for users who wish to create a new account on a website. Its primary function is to collect essential information from users.These input fields include fields for the user's email address,name, surname, password and re-type password.

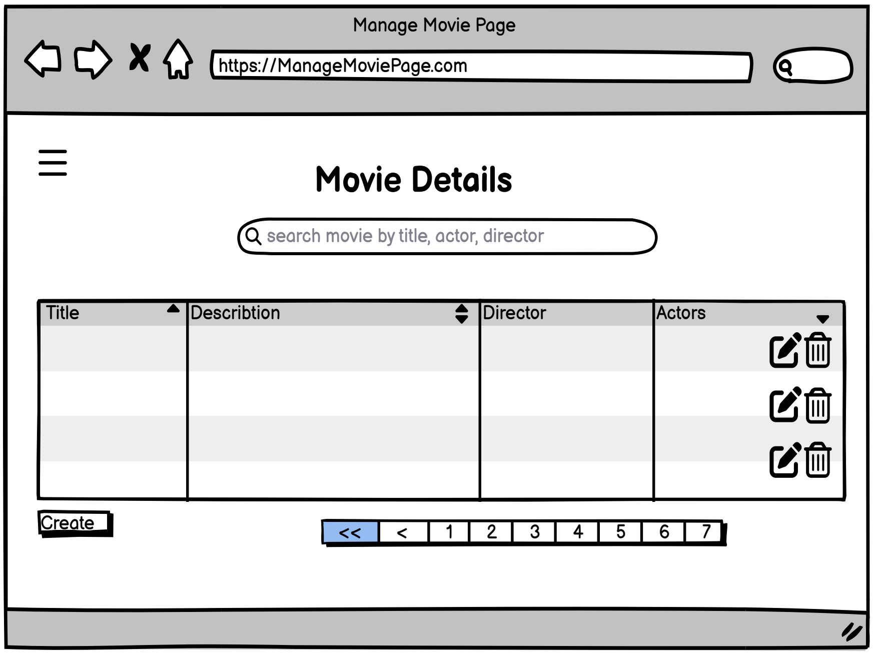
4.3 User Page



This page empowers users with various functionalities:

- Movie Management: Users can effortlessly add or remove movies from their lists. This feature extends to all users, enabling them to curate a personalized selection of movies they plan to watch in the future. It serves as a handy tool for users to maintain a catalog of movies they find intriguing.
- Rating Statistics: Users gain access to insightful statistics concerning movie ratings. This includes valuable data such as the average rating of movies viewed by the user, offering a comprehensive overview of user preferences and popular content.
- User Profile Editing: Users are provided to review and modify their personal details stored on the platform. This encompasses essential information like first name, last name, username, email address, and password.
- Watchlist: Users can maintain a 'Watchlist' within the application, where they compile movies they intend to watch in the future. This feature allows users to easily track and manage their cinematic interests, add new titles, remove ones they lose interest in, and organize the list based on preference.

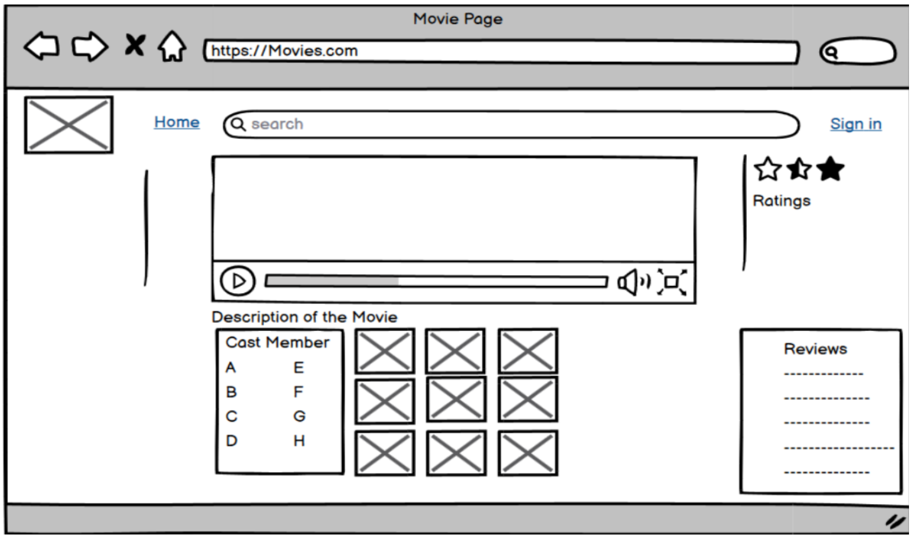
4.4 Admin Page



Add/Delete Movies: This dedicated section empowers administrators to seamlessly manage the application's movie database. Administrators have the capability to add newly released movies or remove outdated entries from the database. The administrator adds the title,director, genre, release data, trailer URL.

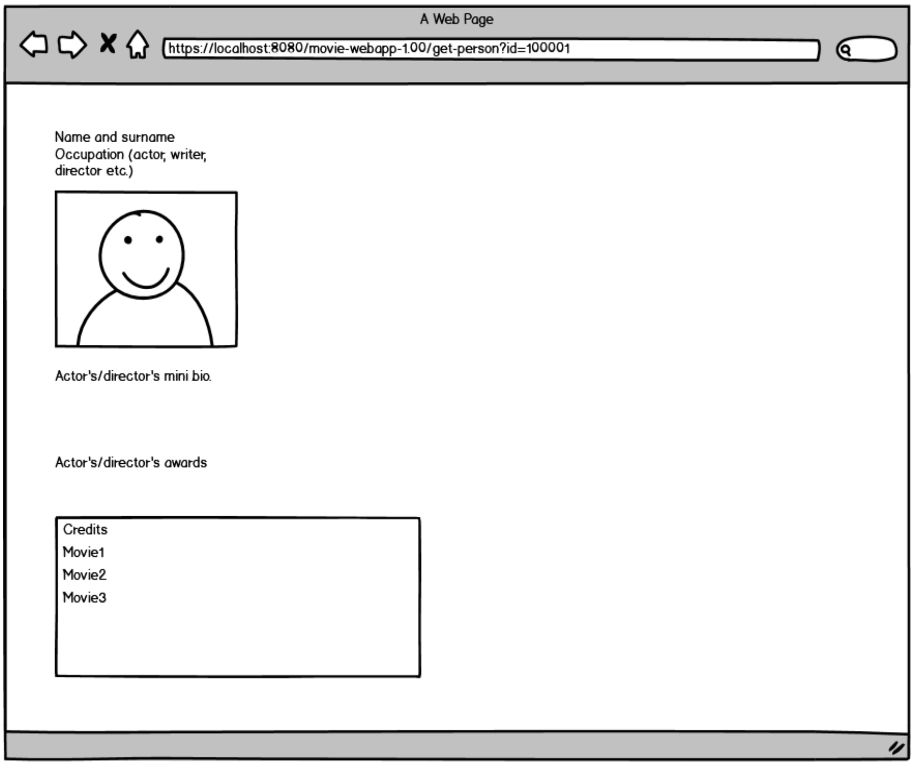
Add/Delete Persons: Within this essential feature, administrators can maintain a comprehensive database of individuals involved in the film industry, including actors, actresses, directors, and writers. Administrators are equipped with a user-friendly form interface to facilitate the addition of new persons to the database. The form typically includes fields such as name, surname, biography, role (e.g., actor, director), birthplace, birthdate, and a link to their photo.

4.5 Movie Page



The book page shows everything you need to know about a movie, like its title, cast member, storyline. Central to the book page's functionality is its interactive nature, which empowers users to actively engage with the movie's content. Users have the opportunity to leave their mark on the narrative by sharing their insights through personalized evaluations and reviews. Through a user-friendly rating system, individuals can express their appreciation or critique of the movie.

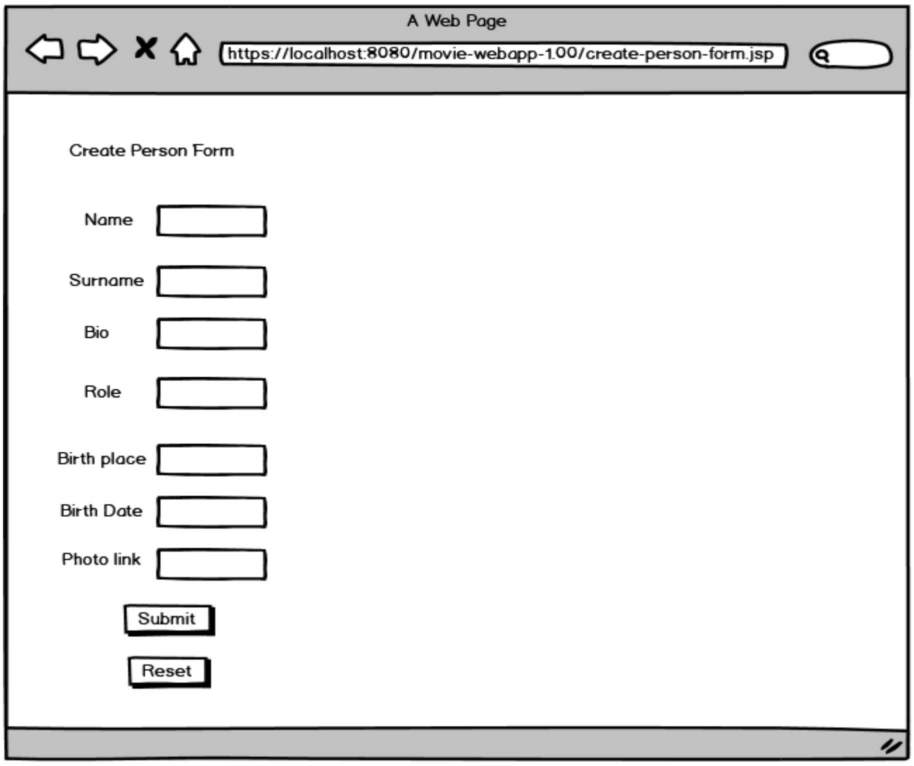
4.6 Persons Page


A Web Page
https://localhost:8080/movie-webapp-1.00/get-person?id=100001
Name and surname
Occupation (actor, writer, director etc.)
Actor's/director's mini bio.
Actor's/director's awards
Credits
Movie1
Movie2
Movie3

This page displays data about a person by their id. It contains:

- Name and Surname: Name and Surname of the person.
- Occupation: Role of the person (director, writer, etc.)
- Photo: photo of the person.
- Short bio: small biography of the person.
- Awards: awards of the person.
- Credits: movies in which the person was involved.

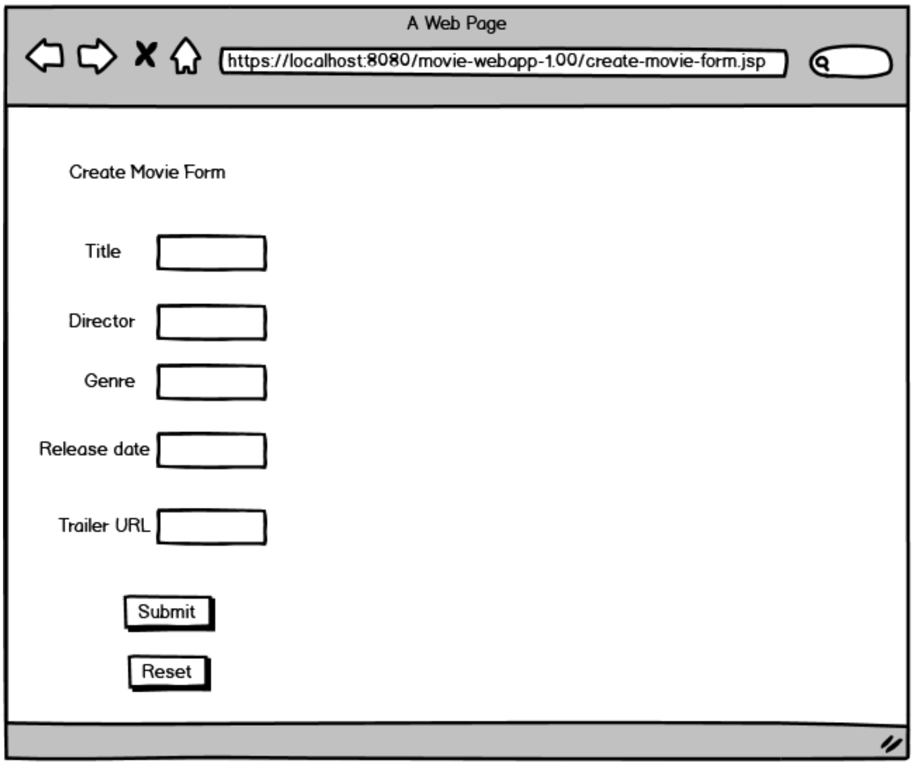
4.7 Add Person Page



This page displays form that user needs to fill to add a person to the database. It contains:

- Name field: Name of the person.
- Surname field: Surname of the person.
- Short bio: Small biography of the person.
- Role: Role of the person (director, writer, etc.)
- Photo link: Link to the photo of the person.
- Birth place: Place of birth of the person.
- Birth date: Date of birth of the person.

4.8 Add Movie Page

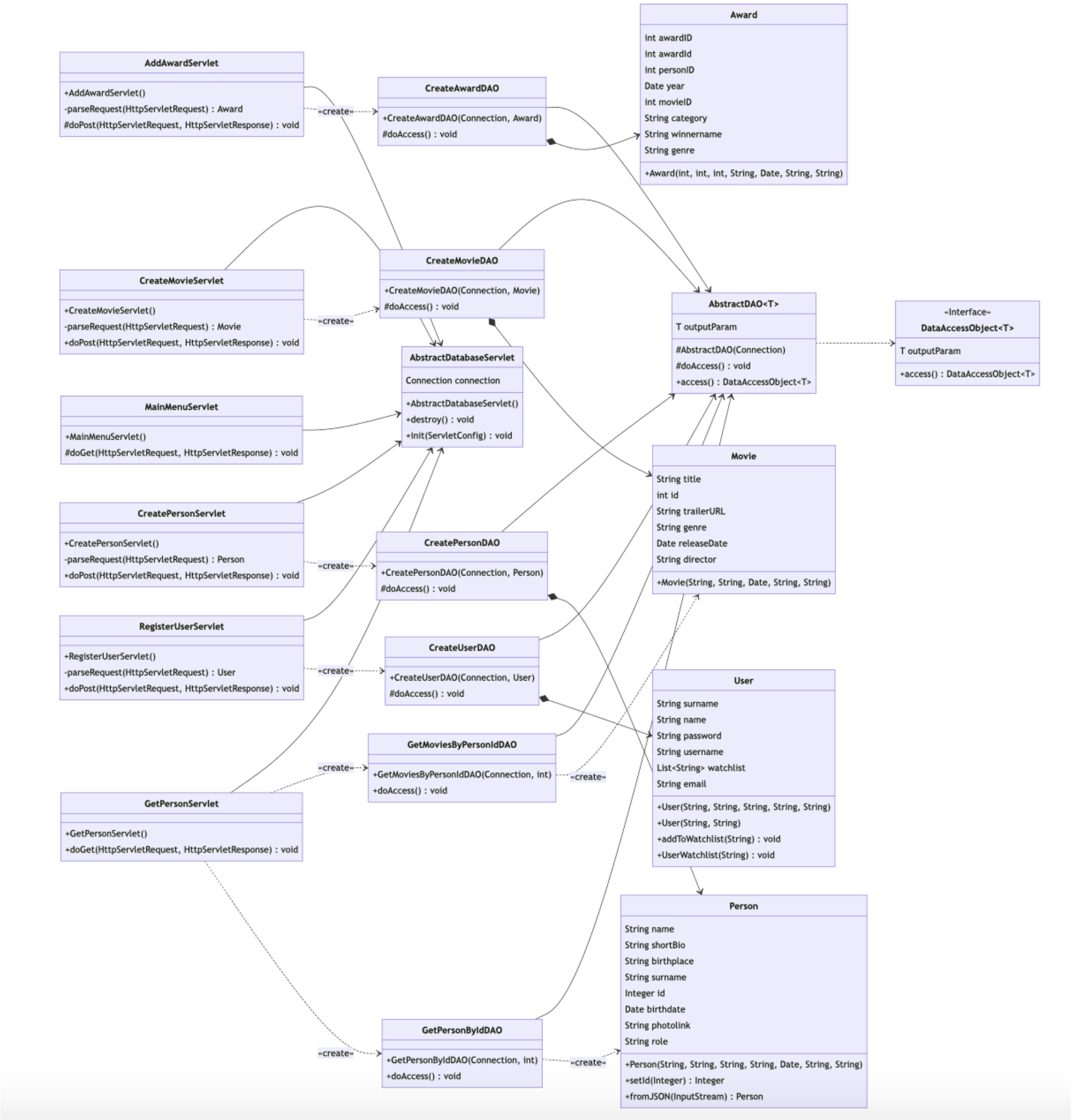


This page displays form that user needs to fill to add a person to the database. It contains:

- Title: Title of the movie.
- Director : Director of the movie.
- Genre: Genre of the movie.
- Release date: Release date of the movie.
- Trailer URL: Link to the trailer of the movie.

5 Business Logic Layer

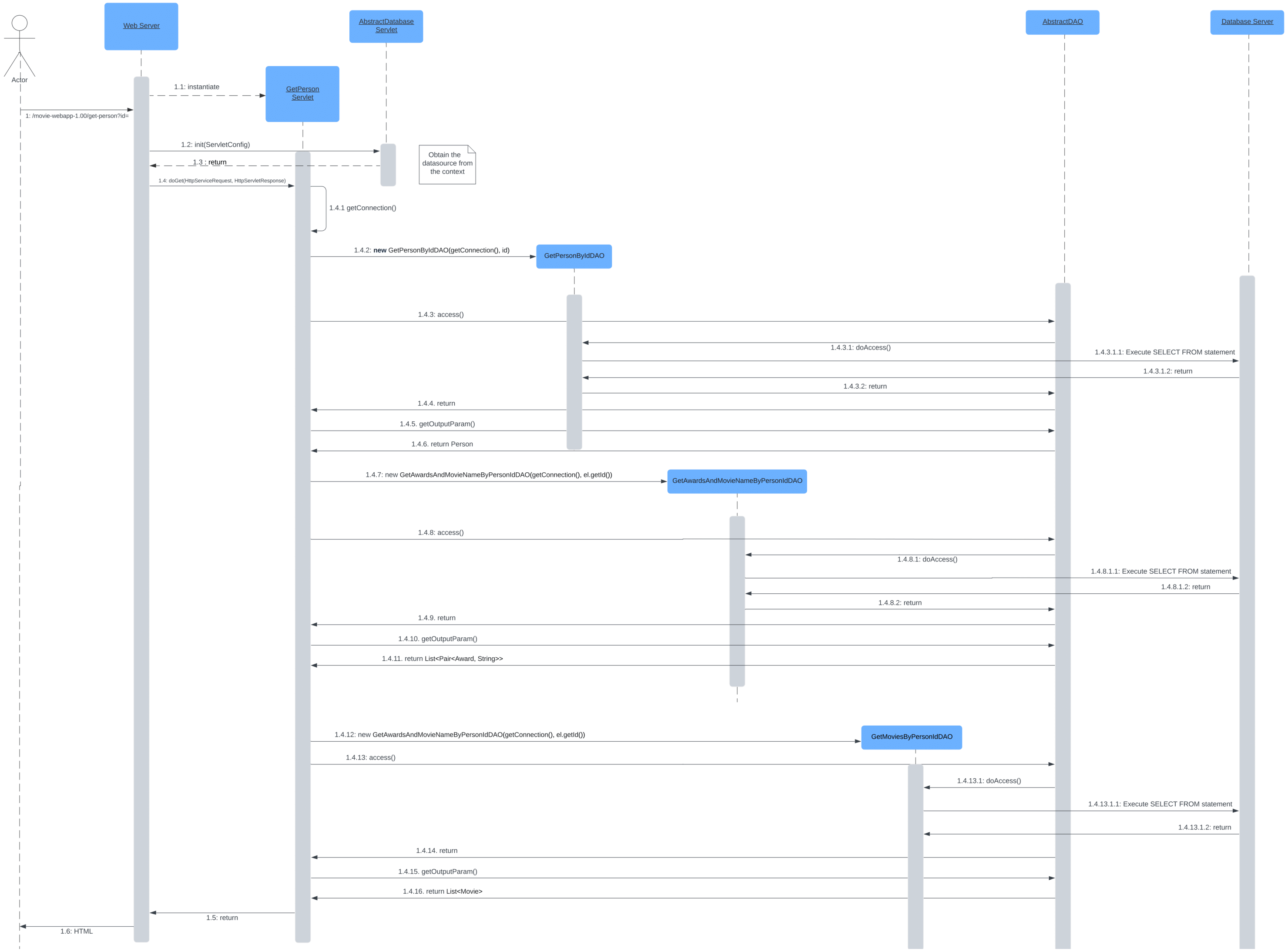
5.1 Class Diagram


AddAwardServlet
+AddAwardServlet()
-parseRequest(HttpServletRequest) : Award
#doPost(HttpServletRequest, HttpServletResponse) : void
«create»
CreateAwardDAO
+CreateAwardDAO(Connection, Award)
#doAccess() : void
Award
int awardID
int awardId
int personID
Date year
int movieID
String category
String winnername
String genre
+Award(int, int, int, String, Date, String, String)
CreateMovieServlet
+CreateMovieServlet()
-parseRequest(HttpServletRequest) : Movie
+doPost(HttpServletRequest, HttpServletResponse) : void
«create»
CreateMovieDAO
+CreateMovieDAO(Connection, Movie)
#doAccess() : void
AbstractDAO<T>
T outputParam
#AbstractDAO(Connection)
#doAccess() : void
+access() : DataAccessObject<T>
«Interface»
DataAccessObject<T>
T outputParam
+access() : DataAccessObject<T>
AbstractDatabaseServlet
Connection connection
+AbstractDatabaseServlet()
+destroy() : void
+init(ServletConfig) : void
MainMenuServlet
+MainMenuServlet()
#doGet(HttpServletRequest, HttpServletResponse) : void
Movie
String title
int id
String trailerURL
String genre
Date releaseDate
String director
+Movie(String, String, Date, String, String)
CreatePersonServlet
+CreatePersonServlet()
-parseRequest(HttpServletRequest) : Person
+doPost(HttpServletRequest, HttpServletResponse) : void
«create»
CreatePersonDAO
+CreatePersonDAO(Connection, Person)
#doAccess() : void
RegisterUserServlet
+RegisterUserServlet()
-parseRequest(HttpServletRequest) : User
+doPost(HttpServletRequest, HttpServletResponse) : void
«create»
CreateUserDAO
+CreateUserDAO(Connection, User)
#doAccess() : void
User
String surname
String name
String password
String username
List<String> watchlist
String email
+User(String, String, String, String, String)
+User(String, String)
+addToWatchlist(String) : void
+UserWatchlist(String) : void
«create»
GetMoviesByPersonIdDAO
+GetMoviesByPersonIdDAO(Connection, int)
+doAccess() : void
«create»
GetPersonServlet
+GetPersonServlet()
+doGet(HttpServletRequest, HttpServletResponse) : void
Person
String name
String shortBio
String birthplace
String surname
Integer id
Date birthdate
String photolink
String role
+Person(String, String, String, String, Date, String, String)
+setId(Integer) : Integer
+fromJSON(InputStream) : Person
«create»
GetPersonByIdDAO
+GetPersonByIdDAO(Connection, int)
+doAccess() : void
«create»

The class diagram contains (some of) the classes used to handle User, Movie, Person, and Award. We describe several servlets: MainMenuServlet which retrieves the information of a movie and awards, and the CreateMovieServlet which handles the Movie resource and allows to creation of a movie. Similarly, CreatePersonServlet handles the Person resource and allows the creation of a person according to the attributes it has. Each servlet extends the AbstractDatabaseServlet which is needed to acquire the connection to the database. The movie resource is handled by two servlets CreateMovieServlet and MoviePageServlet: these servlets implement doPost and doGet methods accordingly. Existing user's login operation is handled by UserLoginServlet and RegisterUserServlet handles the registration operation of a new user. The MovieDAO class interacts with the database to retrieve movie information and CreateMovieDAO creates a new movie resource, while the GetMovieByPersonIdDAO class allows to retrieve the information about a movie given the PersonId (in this case the director, actor, or any other role). Each DAO class extends the AbstractDAO class (which is delegated to the interaction with the database) and defines the doAccess() method. The AbstractDAO class implements the DataAccessObject interface.

5.2 Sequence Diagram

Here reported the sequence diagram for the requesting a person from database. The user executes a GET request to the web server, specifying the URI /movie-webapp-1.00/get-person. Additionally, the data about the person which is trying to retrieve information (personID) is passed to the web server. The web server instantiates the GetPersonServlet and calls its doGet() method, passing the HttpServletRequest and the HttpServletResponse. Given the GET data, it is searching a person inside the database with person ID data. The control is passed to the GetPersonByIdDAO which receives as an argument the connection (defined in AbstractDatabaseServlet extended by the GetPersonServlet) and searched the person. The GetPersonByIdDAO extends the AbstractDAO and contacts the Database Serverwhich executes the SQL statement for searching a person. This DAO return the personal information about the person. To collect all the information about the person, GetPersonServlet performs two other Database access operations; GetAwardsAndMovieNameByPersonIdDAO and GetMoviesByPersonIdDAO. The purpose of these operations is to collect all the information about the person including movies which s/he participated and awards which s/he won. If some errors occur, the control is returned to the GetPersonServlet and a new String object contains the relevant situation is created. Finally, the person found in the database is returned to the user with all the information about the person.


Actor
Web Server
GetPerson Servlet
AbstractDatabase Servlet
AbstractDAO
Database Server
1.1: instantiate
1: /movie-webapp-1.00/get-person?id=
1.2: init(ServletConfig)
1.3 : return
1.4: doGet(HttpServiceRequest, HttpServletResponse)
Obtain the datasource from the context
1.4.1 getConnection()
1.4.2: new GetPersonByIdDAO(getConnection(), id)
GetPersonByIdDAO
1.4.3: access()
1.4.3.1: doAccess()
1.4.3.1.1: Execute SELECT FROM statement
1.4.3.1.2: return
1.4.3.2: return
1.4.4. return
1.4.5. getOutputParam()
1.4.6. return Person
1.4.7: new GetAwardsAndMovieNameByPersonIdDAO(getConnection(), el.getId())
GetAwardsAndMovieNameByPersonIdDAO
1.4.8: access()
1.4.8.1: doAccess()
1.4.8.1.1: Execute SELECT FROM statement
1.4.8.1.2: return
1.4.8.2: return
1.4.9. return
1.4.10. getOutputParam()
1.4.11. return List<Pair<Award, String>>
1.4.12: new GetAwardsAndMovieNameByPersonIdDAO(getConnection(), el.getId())
GetMoviesByPersonIdDAO
1.4.13: access()
1.4.13.1: doAccess()
1.4.13.1.1: Execute SELECT FROM statement
1.4.13.1.2: return
1.4.14. return
1.4.15. getOutputParam()
1.4.16. return List<Movie>
1.5: return
1.6: HTML

5.3 REST API Summary

URI	Method	Description
/rest/person	POST	Creates a new person

Table 2: Describe in this table your REST API

5.4 REST Error Codes

Error Code	HTTP Status Code	Description
-200	OK	OK
-300	BAD_REQUEST	Wrong id format
-301	BAD_REQUEST	One or more input fields are empty
-302	BAD_REQUEST	Email missing
-303	BAD_REQUEST	Password missing
-304	BAD_REQUEST	Submitted credentials are wrong
-305	CONFLICT	Mail already used
-306	BAD_REQUEST	Wrong rest request format
-307	NOT_FOUND	User not found
-308	BAD_REQUEST	The input JSON is in the wrong format
-400	BAD_REQUEST	Operation unknown
-404	NOT_FOUND	Request not found
-405	METHOD_NOT_ALLOWED	The method is not allowed
-500	INTERNAL_SERVER_ERROR	Internal Error

Table 3: Describe in this table your REST API

5.5 REST API Details

A resource

This API is used for inserting new Person to the database.

- URL: `/rest/person`
- Method: `POST`
- URL Parameters: None
- Data Parameters: {"person": {"name":"Name", "surname":"Surname", "shortBio":"Creator of Movie", "birthplace":"New York", "birthdate":"1900-11-11", "photolink":"https://rb.gy/1gr6bf", "role":"Role"}}
- Success Response:
  - Code: 200 OK
- Error Response:
  - Code: 400 BAD REQUEST
  - Code: 500 INTERNAL SERVER ERROR
  - Code: 404 NOT FOUND

- Code: 405 METHOD NOT ALLOWED

Sample Call: curl -v -X POST -H ”Content-Type: application/json” -d {”person”: {”name”:”Name”, ”surname”:”Surname”, ”shortBio”:”Creator of Movie”, ”birthplace”:”New York”, ”birthdate”:”1900-11-11”, ”photolink”:”https://
profile-picture-973460_640.png”, ”role”:”Role”}} http://localhost:8080/movie-webapp-1.00/rest/person

6 Group Members Contribution

Umut Berk CAKMAKCI: Contributed to the ER diagram and explanation, Class diagram, DB physical schema and populating tables, Home page prototype, Sequence diagram explanation, created all of the User-related java classes (resource, DAO, servlets) including .jsp files.

Meltem YANOGLU: Contributed to the ER diagram and explanation, DB physical schema and populating tables, Page prototypes, contributed to the report especially UML Class Diagram explanations and reviewed it, created UML Class Diagram and modified it, Award, Watchlist java classes, and related DAO's and Servers, and.jsp files.

Kejsi BIMAJ: Contributed to the ER diagram and explanation, DB physical schema, Admin and Login/Signup Page prototype, objectives, main functionalities, presentation login layer, created Admin and Rates java classes (resource, DAO, servlets), including .jsp files.

Vladislav TIKHONOV: Created Person page prototype, added initial project, added log4j to pom file, created Person.java, Role.java and CreatePersonDAO.java, created SearchPersonByIdDAO.java, created Webpages Diagram, added database context, created GetPersonServlet.java, created CreateMovieServlet.java and create-movie-form.jsp, created GetAwardsAndMovieNameByPersonIdDAO.java, added REST API implementation, tested and fixed errors in the servlets.

Muhammad ALI Contributed to the development of the ER diagram and DB schema and Explanation. Developed main menu and Movie Page Prototype, DAO files, Servlets and Corresponding JSP files. Wrote Java files to encapsulate complex business logic, enhancing the project's functionality and implementing backend logic to support frontend features seamlessly .