## Architecture

The TravelGlance web application is a CRUD application that employs a client-server architecture. The client-side is primarily responsible for user interaction and presentation, while the server-side handles data storage, authentication, and business logic. The application utilizes a combination of HTML, CSS, JavaScript, and PHP for its implementation.

## Algorithms

Login:

* + Check if the login and password fields have valid lengths.
  + Check if the entered login exists in the users.json file.
  + If login exists, verify the password using the password\_verify() function in PHP.
  + If the password is valid, save the user's login, ID, and avatar file name in the session.
  + Reload the page to display the user's home page.

Signup:

* + Validate the login and password lengths.
  + Check if the entered login already exists using the login\_search.php script.
  + Check if the uploaded avatar file has a valid extension.
  + If all validations pass, create a new user object and save it to the users.json file.
  + Store the user's login, ID, and avatar file name in the session.
  + Reload the page to display the user's home page.

Record Management:

* + Validate the record data, including date, amount, and city.
  + Check if the entered amount is a valid number.
  + Save the new record to the records.json file.

Public Records Page:

* + Fetch the records data from the get-records.php script.
  + Display the records in a paginated manner, using the change-records-page.php script.

Home Page:

* + Fetch the user's records from the get-user-records.php script.
  + Display the user's records and avatar.

Edit Account:

* + Validate the entered login, password, and new login/avatar.
  + Check if the current password is correct using the password-verifying-process.php script.
  + If the validations pass, update the user's data in the users.json file.
  + If the avatar is changed, update the avatar file name in the session.
  + Reload the page to display the updated user's home page.

## Main Files

* index.php: The main entry point of the application, responsible for displaying the initial page (public records page) and routing the user to other pages based on their actions.
* data: The directory containing data files for user accounts, trip records, and user avatars.
* pages: The directory containing HTML files for each page of the application (public records, home, error-page).
* php: The directory containing PHP scripts for various functionalities, including login, signup, record management, authentication, and user data manipulation.
* res: The directory containing application-related resources, such as images, CSS files, and JavaScript files.
* scripts: The directory containing JavaScript scripts for user interactions, validation, and record management.
* styles: The directory containing CSS files for styling the application pages.
* animation.js: Handles state changing animation of the top navigation bar on each page.
* authentication-authorisation.js: Contains scripts that are related with login, sign-up, logout, and edit account processes.
* validation.js: Contains all validation related functions.
* record-model.js: Handles all functionality related to the record managing.
* public-records.js: Operations only on the public records page.
* home.js: Operations only on the home page.
* index.js: Scripts that can be used on all pages and in all other JS files.
* login-change-checking.php: PHP script that returns boolean value to the JS fetch method to check out if the new entered user login in the edit account window is available.
* edit-account-data-process.php: Php script that get users new data entered in the edit account window and changes user data to the new once, if the validation passes.
* sign-up-process.php: PHP script that handles the signup process.
* log-in-process.php: PHP script that handles the login process.
* get-records.php: PHP script that returns a part of all users' records.
* change-records-page.php: PHP script that handles changing page of the displayed part of records.
* making-record-process.php: PHP script that handles the process of making a new record.
* deleting-record-process.php: PHP script that handles the process of deleting a record.
* login-searching-process.php: PHP script that returns a boolean value based on if the passed login to it where found in the users.json file.
* get-user-records.php: Peach be script that returns a list of all user records.
* login-search.php: Contains the function that checks if the passed login already exists in the users.json file.
* deleting-account-process.php: PHP script that handles the process of deleting a user account.
* delete-user-records.php: PHP script that handles deleting of all user related records.
* get-the-user-login-avatar.php: PHP script that return the list of objects that encapsulate user login and avatar file name to display it by each record on the public records page.
* verify-login-password.php: contains the function to find the passed user login and in case of success verify the passed password with that one in the user.json file.
* password-verifying-process.php: PHP script that checks if the entered current password in the edit account window is right.
* Record.php: Class for the record object that is displayed on the home page.
* DetailedRecord.php: Class for the detailed record object that is displayed on the public records page.
* User.php: Class for the user object.
* UserLoginAvatar.php: Class for the object that encapsulates the user record and avatar file name.
* UserDataCheck.php: Class that is used for her return object value for the login process PHP script.
* log-out-process.php: PHP script that handles the log out process.
* get-user-from-list.php: contains the function that returns the user from the passed users list by their ID.