

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	CycleK	
Last Name	First Name	Badge Number
Abedini	Kimia	2090249
Boscolo Bacheto	Martina	2109755
Cocco	Alessio	2087635
Munerotto	Giacomo	1217865
Tomaiuoli	Marco	2120827
Trevisiol	Riccardo	2095666

1 Objectives

Our project (Fitness Tracker) aims to develop a Web Application for fitness enthusiasts, to let them have everything they need in one place. Our system will be used to generally store all you need when you start your journey in the gym, going from your diet and your exercises schedule or your personal graph to keep track of your progress, to simply communicate with other passionate users to share advice and talk about anything you want gym related.

2 Main Functionalities

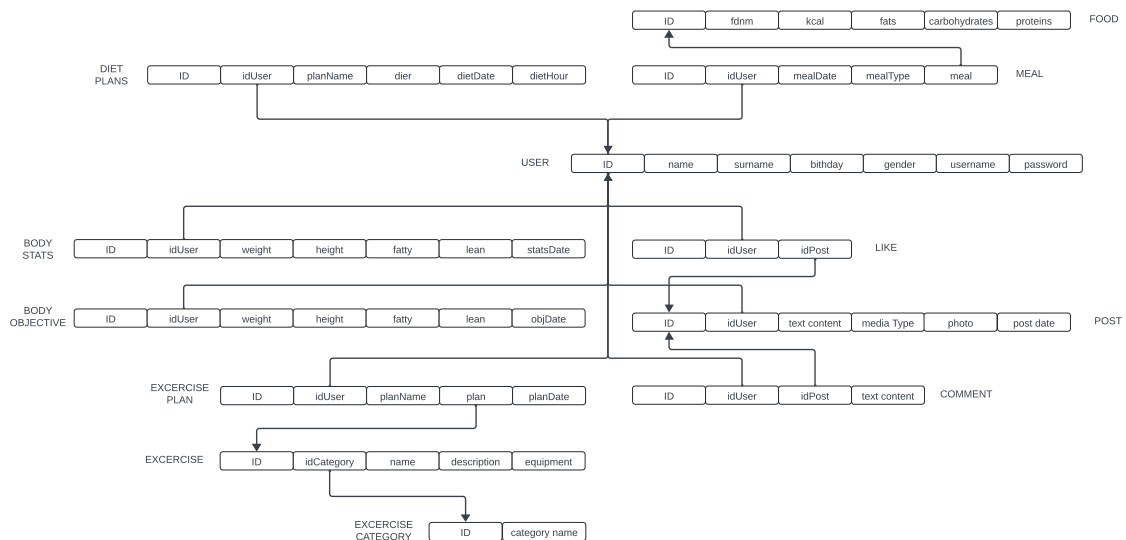
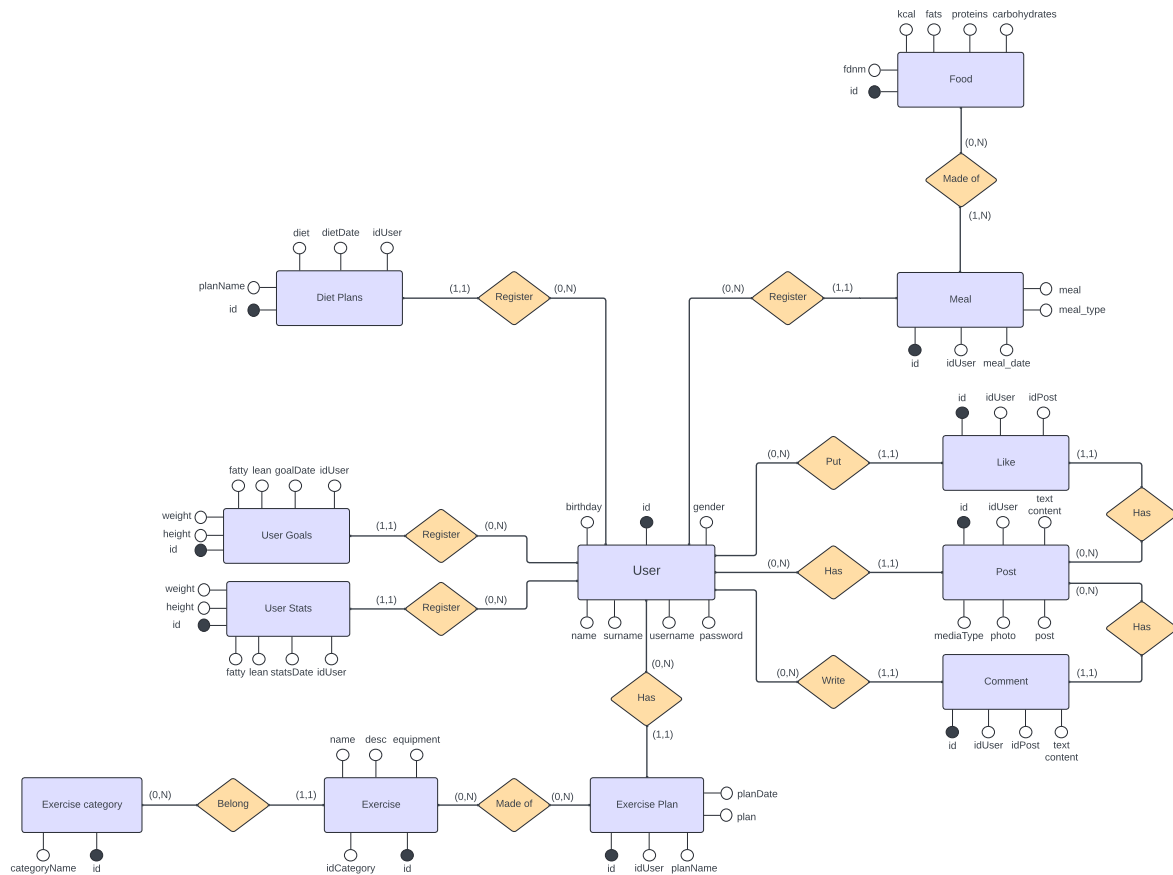
Our web application defines a Fitness Tracker, capable of managing the creation of your personal account, save your exercises schedule with the possibility of adding a description to keep track of your progress. It also allows you to manage your diets with the possibility of creating one or more personalized ones or simply uploading those from your nutritionist. Other important features are meal management, the analysis of your personal statistics with graphs to show your progress(weight loss or gain) and the possibility of communicating in a dedicated forum with the aim of helping and bringing closer to this world the novices.

In particular, our Web Application is divided in 6 main parts:

- User Management: Users can register, login and visualize/modify his personal information.
- Statistics Management: User's Collected data are used to generate an overview of their progress in both Gym and Diet areas.
- Social Network Management: User can make a post, made of text and eventually an image, expressing his gym progress, his current diet or anything else he want. User can also like and comment other users post. An home page will show all the posts with related likes and comments.
- Diets Management: Users can view their diet plans or add multiple ones ensuring the possibility of keeping track of what you have to eat each day.
- Meals Management: Users can view what they ate, can register their meals and can register new foods.
- Exercises Management: Users can see their exercise plans or add their own exercise plans , modify them or remove them.

3 Data Logic Layer

3.1 Entity-Relationship Schema



- **User:** Each user of the application is identified by an ID (type of INT). For each user we also save their Name, Surname, gender, username and password as VARCHAR and birthday of type DATE. The primary key is the ID and we have the constraint UNIQUE for the username field.
- **User Stats:** Contains all records of user's body statistics measures of weight,height, lean and fatty mass. The primary key is the ID but have a unique constraint on idUser and Date that allows the user to insert at most one measures per day.
- **User Goals:** Contains the same field and constraint of the User Stats table but is intended to keep track of user's goals.
- **Posts:** Contains all the posts of the social network. Each post belongs to a specific user and is made of text and eventually a photo and characterized by the timestamp of it's creation. Photos are saved in the DB as Byte type. Primary key is the ID.
- **Likes:** Contains the all the likes for the posts of the social network, each one characterized by the user id of the user who liked the post and the id of the post liked. Constraint on user Id and postId allows user to create only one like per post. Primary key is the ID.
- **Comments:** Contains all the comments, made of text, of the social network, each one characterized by the id of the user who commented the post and the id of the post commented. Primary key is the ID.
- **Diet Plans:** Contains all informations about the diet of the user: planName, dietDate and the entire diet. The primary key is the ID but have a unique constraint on idUser and dietDate that allows the user to change a diet within the first 24 hours of adding it-
- **Meal:** Contains the records of the meals of the user. The primary key is the ID, but has a unique constraint on idUser, date of the meal and the type of the meal: a user can have a type of meal once a day (one breakfast, one lunch, etc). In addition to these fields, the table contains the string with the single meal.
- **Food:** Contains the list of the foods registered. The primary key is the ID. The other fields are the stats for 100g of the food.
- **Exercise Plan:** Contains all the exercises of the user which are: planName, planDate and the entire plan (which is the exercise plans of the user). The primary Key is the ID.
- **Exercise:** Contains the list of the exercises. The primary key is the ID. it has idCategory as a foreign key and also shows the name of the exercise and a description about it and the equipment that are needed for doing the exercise.
- **Exercise Category:** Contains the list of the exercise categories. the primary key is the ID.

3.2 Other Information

We have followed some rules during the implementation of the Database:

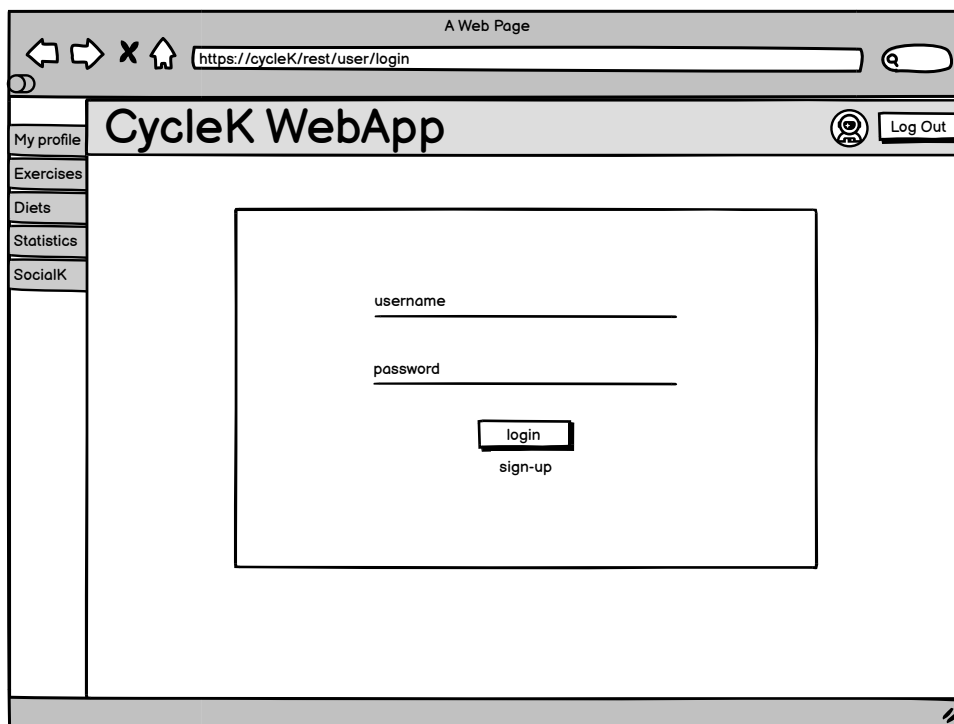
- Every Table must have the ID as the primary key.
- If a table need to reference to another table should have an ID key that point to that table.
E.g.: The User Stats table that contains users entries for body measures must keep track the user who inserted the measure. The UserId refers as foreign key the User table.
- Other constraints can be made during the table creation.
E.g: In the User Stats Table an user can insert only a measure per day

4 Presentation Logic Layer

The web application is divided in the following pages:

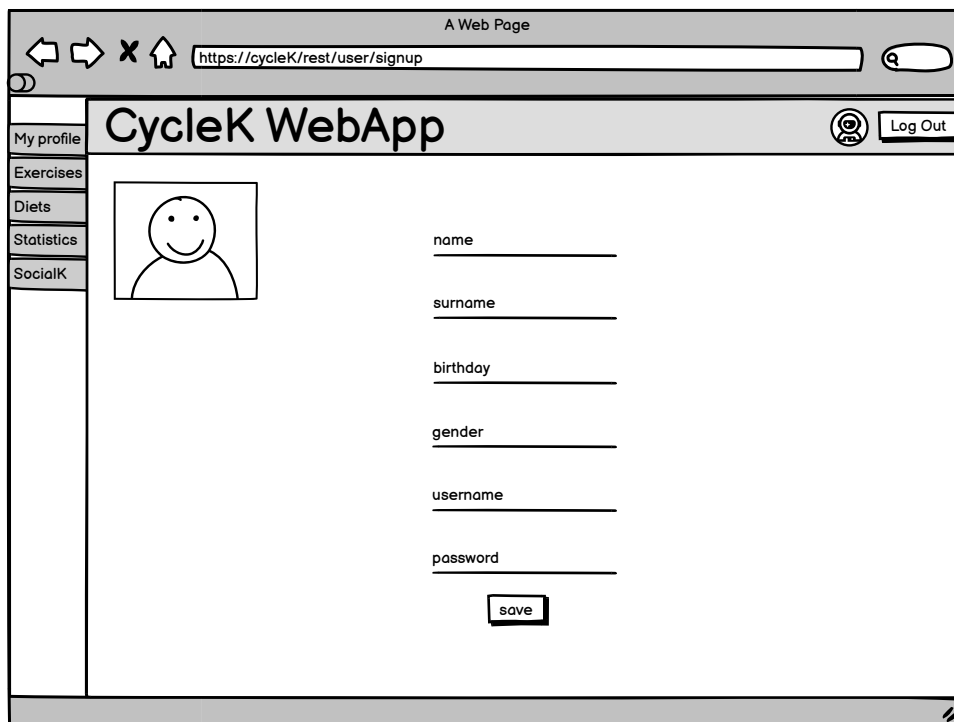
- Homepage: HomePage of the webApp
- LoginPage: Page in which a user can login by providing his/her own credentials.
- UserPage: After a successful login, it's possible to enter into the personal area which contains all the user data.
- Stats Page: An overview of users statistics that includes Body Measures, Goals, Diet and Exercise.
- Social Network Page: A simple social network page where the user can see other users post, like or comment those posts or publish his own post.
- Diet Page: An overview of user's diets which can be modified if int he 24h range simply create a new one
- Meal Page: It's possible to see an overview of the meals consumed by users in various days. The user can register a meal through the button.
- Exercise Page: An overview of the user's exercises which can be modified by the user or he can add a new one.

4.1 Pages mockup



The mockup shows a web browser window titled "A Web Page" with the address bar displaying "https://cycleK/rest/user/login". The page header features the "CycleK WebApp" logo and a "Log Out" button. A sidebar on the left contains navigation links: "My profile", "Exercises", "Diets", "Statistics", and "SocialK". The main content area is a login form with fields for "username" and "password", a "login" button, and a "sign-up" link.

Login Page



A Web Page

https://cycleK/rest/user/signup

CycleK WebApp

My profile

Exercises

Diets

Statistics

SocialK

Log Out

name

surname

birthday

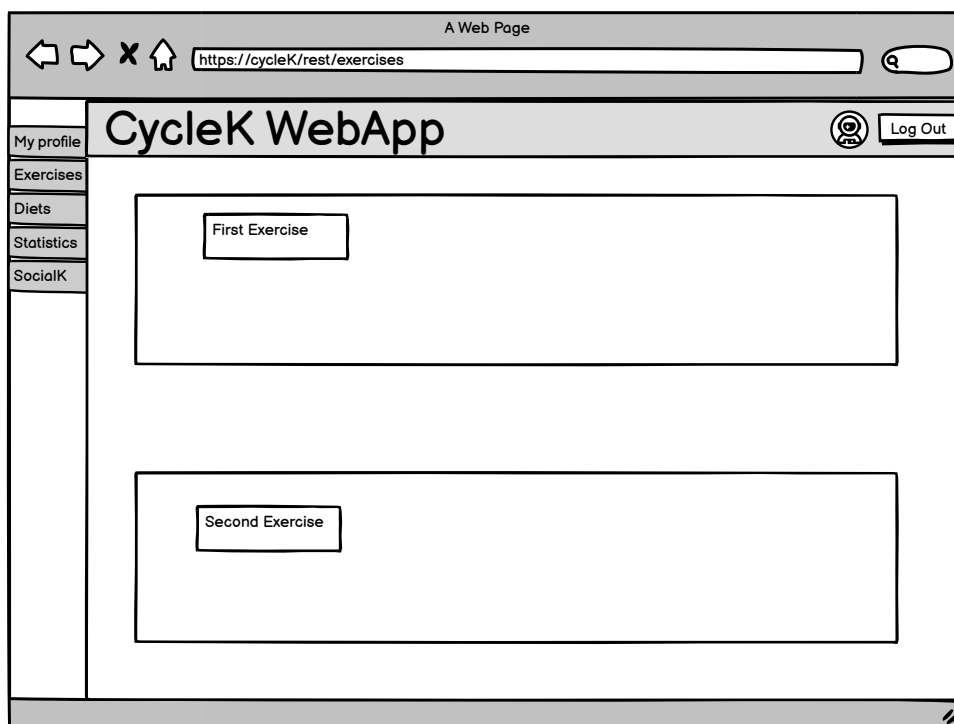
gender

username

password

save

User Profile Page



A Web Page

https://cycleK/rest/exercises

CycleK WebApp

My profile

Exercises

Diets

Statistics

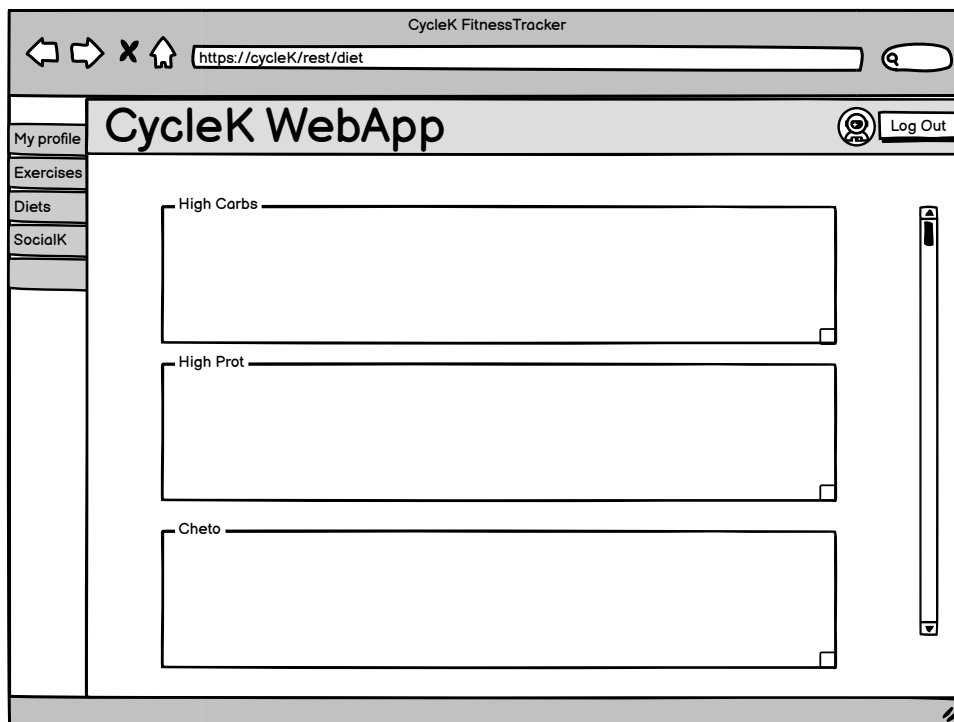
SocialK

Log Out

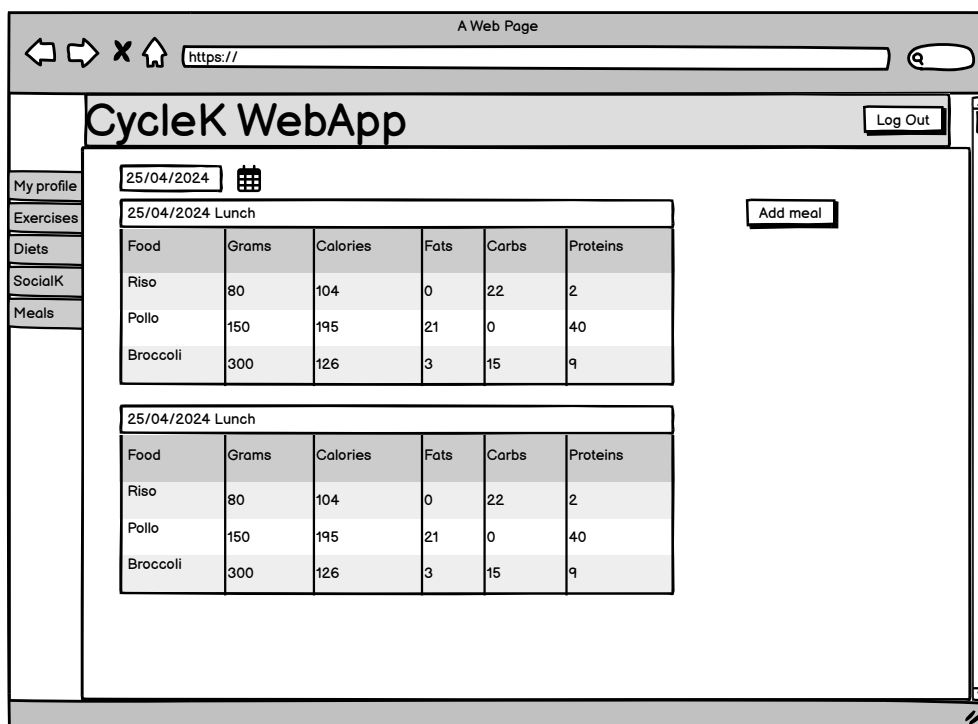
First Exercise

Second Exercise

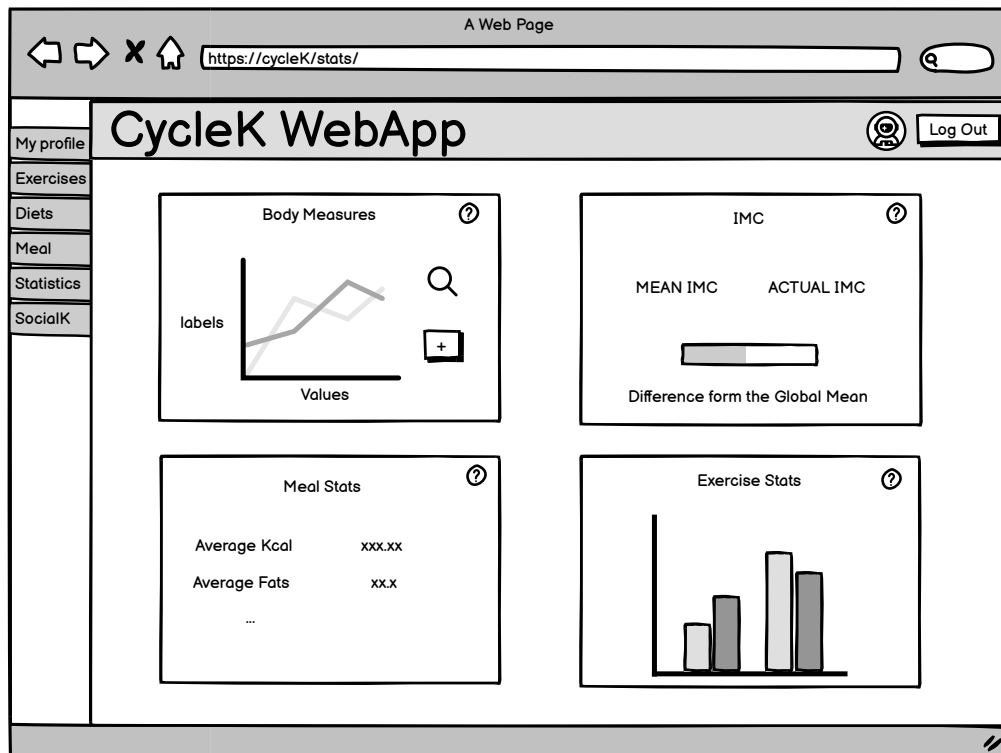
Exercise Page



Diet Page



Meals Page

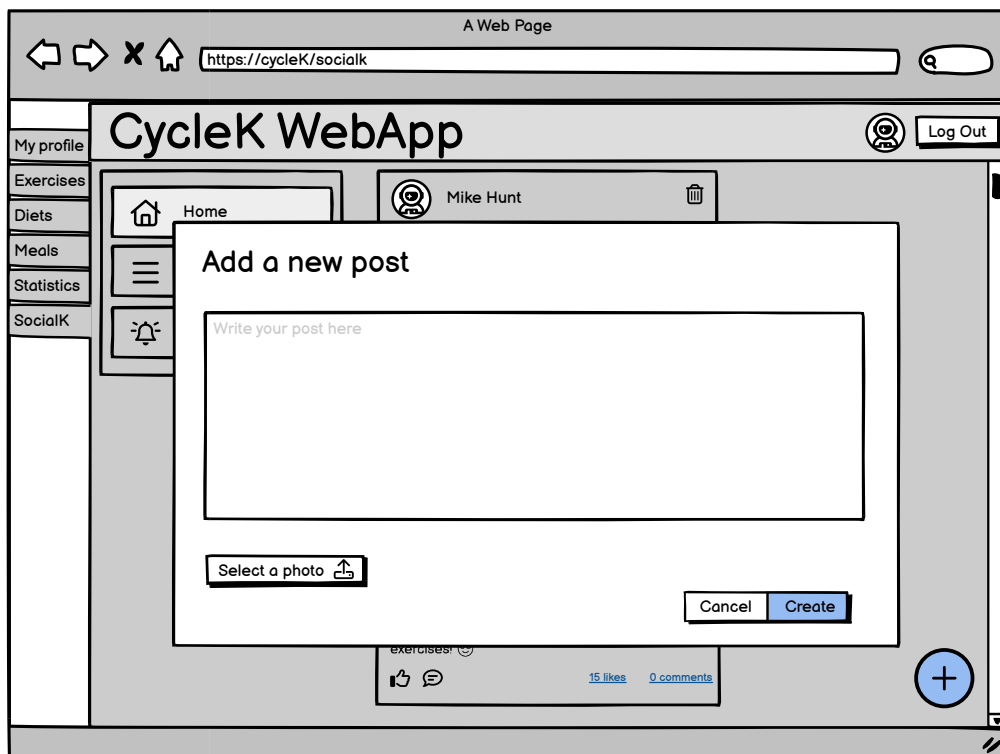


Stats Page

In this Page the user have a general overview of its progress in all fields. A chart will show a comparison among all measures the user inserted with the apposite button, this chart allows the tracking of gain and loos in weight, fatty and lean mass. Another Chart will show the User current and mean IMC computed with its measures and the metrics is compared with the global mean IMC, this allows the user to know his place with respect to other people that are using our app. Other two charts will show some data obtained from the user's meals and exercise to show the average food metrics intake like proteins and other and some exercise metrics. This charts should help the user to understand whether they are flexible or not about diet and exercise.



SocialK home page



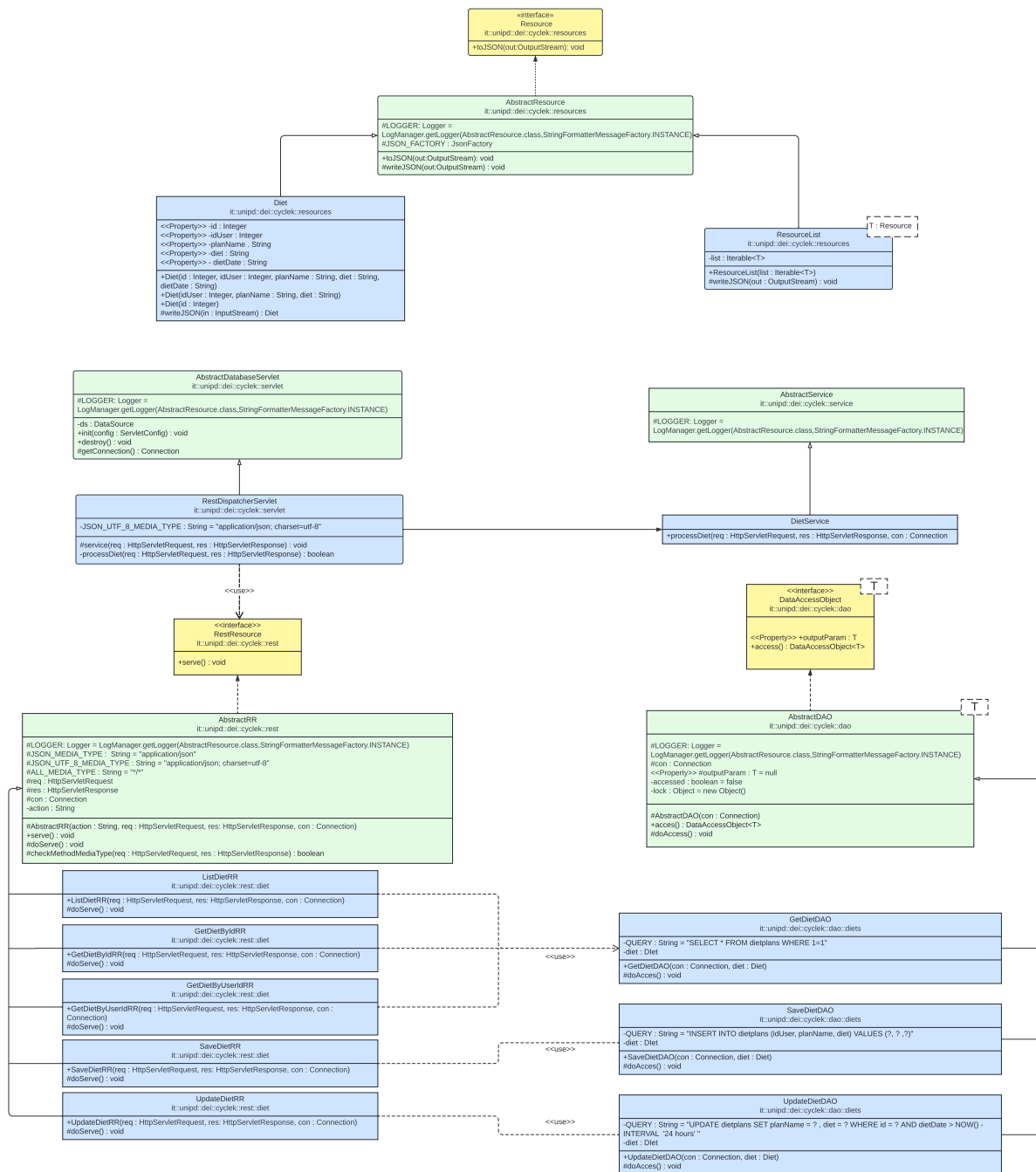
Creation of a new post



Creation of a new comment

5 Business Logic Layer

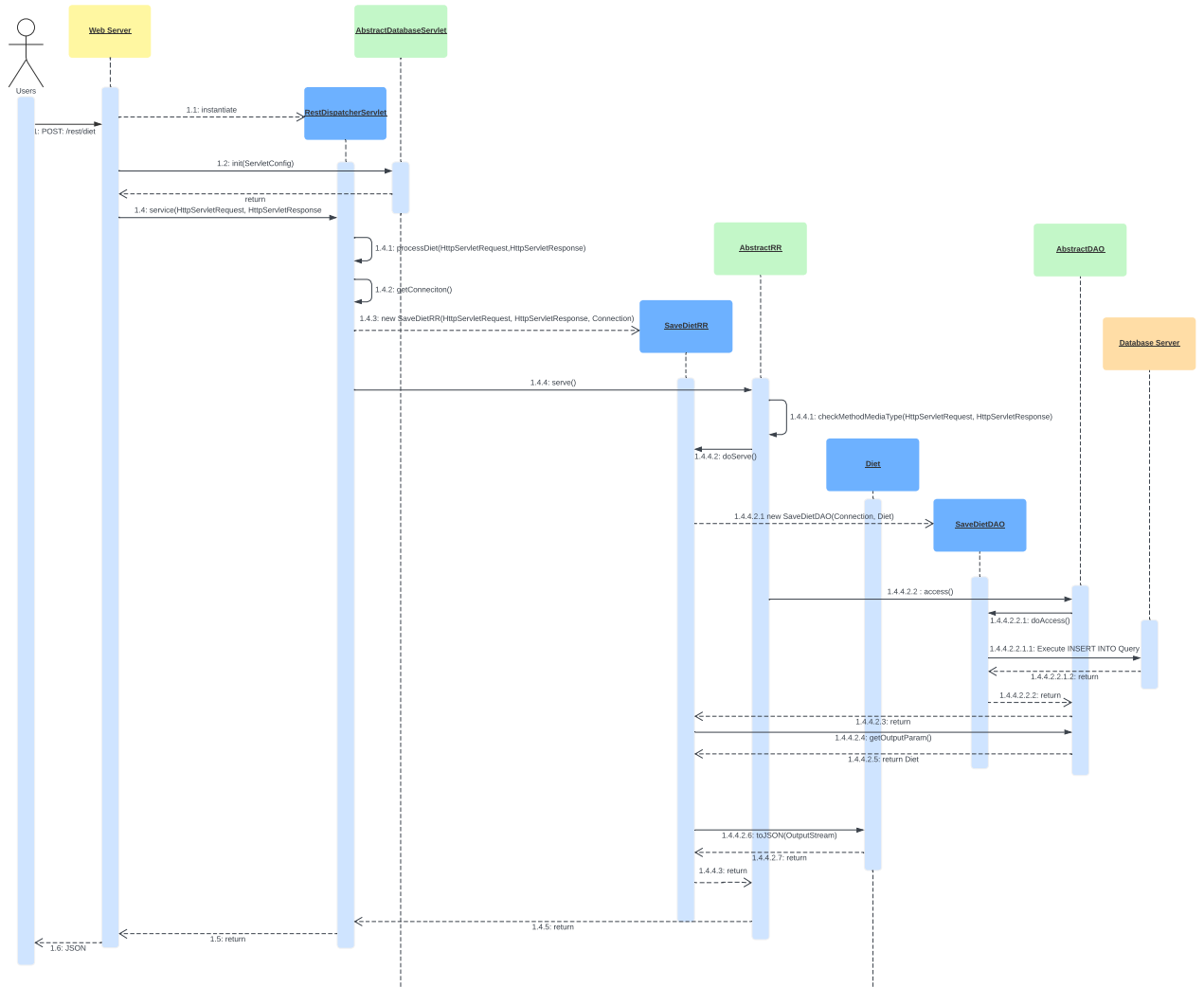
5.1 Class Diagram



The Class Diagram above depicts the diets resource. In the class diagram above, we can see the classes used to handle diet: creation, update, and loading. We have a resource called Diet which implements the constructors and the get methods for the parameters (which corresponds to the parameters in the ER schema). There is a *RestDispatcherServlet* servlet that calls the corresponding rest resource (for example, if the request is a GET for user-specific diets, the *RestDispatcherServlet* calls *GetDietByUserIdRR*). Each one of these resources retrieves the parameters passed in *JSON* format. After doing this, the rest resource calls the *DAO* for the requested method. Then, the *DAO* executes the *SQL* statement and gets the data from the database. Then the *DAO* returns the values requested.

The *DAOs* for the creation of resources implement a *POST* request; for update implement a *PUT* request; finally, the *DAOs* for getting all diets and for getting the diet associated with the id, implement a *GET* request. All the resources, are developed using *REST*.

5.2 Sequence Diagram



Description: In the schema above is shown the sequence diagram for the saving a diet operation. The user sends a *POST* request to the web server, specifying the URI */rest/diet* and the diet in the body of the request. The web server calls the *RestDispatcherServlet*, which calls the right service. The service (in this case *DietService*) analyzes and parses the URI and calls the right Rest Resources (in this case *SaveDietRR*). The resource create a new object of class *Diet*, with the parameters passed in input. After doing this, the resource instantiate a new object of the class *SaveDietDAO*, which extends *AbstractDAO* connecting to the database using the *doAccess()* method. After the connection is correctly established, the DAO executes the *SQL* query and the diet is added to the database.

5.3 REST API Summary

URI	Method	Description
rest/user	GET	Return a JSON containing all the users present in the user table.
rest/user/id/*	GET	Return a JSON containing the user filtered by user id.
rest/user/signup	POST	Insert a new user into the table users, checking that it is not already present. It takes as input a JSON containing all user's information.
rest/user/login	POST	Return the authentication token if the user is present into the table users, otherwise return an error JSON format message. It takes as input a JSON containing the username and the password of the user
rest/user/id/*	PUT	Update information about the given user. It takes a JSON containing all users's information.

Table 2: users REST API

URI	Method	Description
stats/body	GET	Return a list containing all rows of the table stats
stats/body	POST	Insert a new user stats into the table stats
stats/user/{id}	GET	Return a list containing all rows of the table stats filtered by user id
stats/imc/mean	GET	Return the global mean IMC, computed among all latest body measures for each user
stats/imc/user/{id}	GET	Return the actual IMC value and the mean among all measures of a single user
stats/meal/user/{id}	GET	Return stats computed from table meals for a single user
goals	GET	Return a list containing all rows of the table goal
goals	POST	Insert a new body objective into the table goal
goals/user/{id}	GET	Return a list containing all body goal of a single user

Table 3: stats REST API

URI	Method	Description
rest/post	GET	Return a JSON containing all the posts present in the posts table.
rest/post	POST	Create a new post using the JSON passed in the request body
rest/post/{id}	DELETE	Delete the post with given id from the posts table
rest/post/{id}	PUT	Update the post with given id and parameters passed and the request body
rest/post/user/{userId}	GET	Return a JSON containing all the posts for a specific user present in the posts table
rest/post/{id}/like	GET	Return a JSON containing all the likes for the post
rest/post/like	POST	Create a new like associated to that post
rest/post/like/{likeId}	DELETE	Delete the like associated to that post
rest/post/{id}/like/count	GET	Return the number of like for that post
rest/post/comment	POST	Create a new comment associated to that post
rest/post/{id}/comment	GET	Return a JSON containing all the comments for the post
rest/post/comment/{commentId}	DELETE	Delete the comment associated to that post
rest/post/{id}/comment/count	GET	Return the number of comments for that post

Table 4: posts REST API

URI	Method	Description
rest/diet	GET	Return a JSON containing all the diets present in the dietplans table.
rest/diet/id/*	GET	Return a JSON containing the diet filtered by user id.
rest/diet/idUser/*	GET	Return a JSON containing all the diets filtered by idUser
rest/diet	POST	Insert a new diet into dietplans. It takes as input a JSON containing all information about the diet.
rest/diet	PUT	Update a diet based on id passed in the JSON file. Diets can be modified for the first 24 hours after creating them

Table 5: dietplans REST API

URI	Method	Description
rest/foods	GET	Return a JSON containing all the foods with the relative properties of 100g present in the foods table.
rest/foods/id/*	GET	Return a JSON containing the nutritional properties by food id.
rest/foods	POST	Insert a new food into foods table. It takes as input a JSON containing all information about the food.

Table 6: foods REST API

URI	Method	Description
rest/foods	GET	Return a JSON containing all the foods with the relative properties of 100g present in the foods table.
rest/foods/id/*	GET	Return a JSON containing the nutritional properties by food id.
rest/foods	POST	Insert a new food into foods table. It takes as input a JSON containing all information about the food.

Table 7: meals REST API

URI	Method	Description
rest/exercises	GET	Return a Json containing all of the exercises in the exercise table
rest/exercises/id	GET	Return a Json containing the exercise of the specific id in the exercise table
rest/exercise_plan	GET	Return a Json containing all of the exercise plans in the exercise_plan table
rest/exercise_plan/id	GET	Return a Json containing the exercise plan of the specific id in the exercise_plan table
rest/exercise_plan	POST	Insert a new plan into exercise_plan table. It takes as input a JSON containing all information about the exercise plan.
rest/exercise_plan/id	PUT	Update a plan based on id passed in the JSON file.
rest/exercise_plan/id	DELETE	DELETE a plan based on id passed in the path.

Table 8: exercise REST API

5.4 REST Error Codes

We reserved a slot of 100 Error code for each service:

Error Code	HTTP Status Code	Description
-100	400: Bad Request	Missing some user's fields
-101	500: Internal Server Error	Error creating user
-102	400: Bad Request	Username already used
-103	500: Internal Server Error	Unexpected Error while registering user
-104	400: Bad Request	Missing username or password
-105	404: Not Found	User Not Found
-106	500: Internal Server Error	Unexpected Error while login user
-107	400: Bad Request	Cannot update the user, body must contains all parameters
-108	500: Internal Server Error	Unexpected Error while updating user
-200	500: Internal Server Error	Unexpected Error while creating a user goal
-201	409: Conflict	Cannot create a user goal, already exist
-202	400: Bad Request	Cannot create a user goal, body must contains all parameters
-203	400: Bad Request	No User Goal JSON object found in the request
-204	500: Internal Server Error	Unexpected Error while retrieving goals
-205	404: Not Found	No goal found
-206	500: Internal Server Error	Unexpected Error while retrieving a user goal
-207	404: Not Found	No goal found for searched user
-208	500: Internal Server Error	Unexpected Error while creating a user stats
-209	409: Conflict	Cannot create a user stats, already exist
-210	400: Bad Request	Cannot create a user stats, body must contains all parameters
-211	400: Bad Request	No User Stats JSON object found in the request
-212	500: Internal Server Error	Unexpected Error while retrieving stats
-213	404: Not Found	No stat found
-214	500: Internal Server Error	Unexpected Error while retrieving user stats
-215	404: Not Found	No stat found for searched user
-216	500: Internal Server Error	Unexpected Error while retrieving IMC
-217	404: Not Found	IMC cannot be computed, no stats found for searched user
-218	500: Internal Server Error	Unexpected Error while retrieving mean IMC
-219	404: Not Found	Mean IMC cannot be computed, no stats found
-220	500: Internal Server Error	Unexpected Error while retrieving meal stats
-221	404: Not Found	Cannot compute meal stats, no meal registered for searched user
-300	409 : Conflict	Post already exists
-301	500 : Internal Server Error	Unexpected error while creating post
-302	400 : Bad Request	No Post JSON object found in the request
-303	500 : Internal Server Error	Unexpected db error while creating post
-304	404 : Not Found	Post not found
-305	400 : Bad Request	Cannot delete the post, wrong format for URI /post/{postId}
-306	409 : Conflict	Cannot delete the post, other resources depend on it
-307	500 : Internal Server Error	Unexpected db error while deleting post
-308	500 : Internal Server Error	Unexpected error while listing posts
-309	500 : Internal Server Error	Unexpected db error while listing posts
-310	404 : Not Found	Post not found
-311	400 : Bad Request	Cannot get the post, wrong format for URI /post/{postId}
-312	500 : Internal Server Error	Unexpected db error while getting post
-313	400 : Bad Request	Cannot update the post, URI request and post resource postId differ

-314	404 : Not Found	Post not found
-315	400 : Bad Request	Cannot update the post, wrong format for URI /post/{postId}
-316	400 : Bad Request	Cannot update the post, no Post JSON object found in the request
-317	500 : Internal Server Error	Unexpected db error while updating post
-318	500 : Internal Server Error	Unexpected error while counting comments
-319	500 : Internal Server Error	Unexpected db error while counting comments
-320	500 : Internal Server Error	Unexpected error while creating comment
-321	400 : Bad Request	No Comment JSON object found in the request
-322	409 : Conflict	Comment already exists
-323	500 : Internal Server Error	Unexpected db error while creating comment
-324	404 : Not Found	Comment not found
-325	400 : Bad Request	Cannot delete the comment, wrong format for URI /post/comment/{commentId}
-326	409 : Conflict	Cannot delete the comment, other resources depend on it
-327	500 : Internal Server Error	Unexpected error while deleting comment
-328	500 : Internal Server Error	Unexpected error while listing comments
-329	500 : Internal Server Error	Unexpected error while counting likes
-330	500 : Internal Server Error	Unexpected db error while counting likes
-331	500 : Internal Server Error	Unexpected error while creating like
-332	400 : Bad Request	No Like JSON object found in the request
-333	500 : Internal Server Error	Unexpected db error while creating like
-334	500 : Internal Server Error	Unexpected error while deleting like
-335	400 : Bad Request	Cannot delete the like, wrong format for URI /post/like/{likeId}
-336	409 : Conflict	Cannot delete the like, other resources depend on it
-337	500 : Internal Server Error	Unexpected db error while deleting like
-338	500 : Internal Server Error	Unexpected error while listing likes
-339	500 : Internal Server Error	Unexpected db error while listing likes
-400	404: Not Found	Id not found for searched diet
-401	500: Internal Server Error	Unexpected error while retrieving diet
-402	404: Not Found	idUser not found for searched diet
-403	500: Internal Server Error	Unexpected error while retrieving diet
-404	404: Not Found	There are no diets to list
-405	500: Internal Server Error	Unexpected error while listing diets
-406	500: Internal Server Error	Error while saving diet
-407	400: Bad Request	No idUser JSON object found in the request
-408	400: Bad Request	Cannot modify a diet, 24 hours has passed
-409	500: Internal Server Error	Unexpected error while updating a diet
-500	404: Not Found	No food found for the id
-501	500: Internal Server Error	Unexpected Error while retrieving food for this id
-502	404: Not Found	Food not found in database
-503	500: Internal Server Error	Unexpected Error while retrieving food
-504	400: Bad Request	No JSON object found in the request
-505	500: Internal Server Error	Unexpected Error while saving food
-506	404: Not Found	idUser not found for searched meals
-507	500: Internal Server Error	Unexpected Error while retrieving meals
-508	404: Not Found	There are no meals to list
-509	500: Internal Server Error	Unexpected Error while listing meals
-510	400: Bad Request	No JSON object found in the request
-511	500: Internal Server Error	Unexpected Error while saving meal

-600	404: Not Found	id exercise not found
-601	500: Internal Server Error	Unexpected Error while retrieving exercise
-602	404: Not Found	There are no exercises to list.
-603	500: Internal Server Error	Unexpected Error while listing exercises.
-604	400: Bad Request	No JSON object found in the request.
-605	500: Internal Server Error	Unexpected Error while adding exercise plan.
-606	500: Internal Server Error	Unexpected Error while getting exercise plan.
-607	404: Not Found	id exercise plan not found.
-608	404: Not Found	There are no exercise plans to list.
-609	500: Internal Server Error	Unexpected Error while listing exercise plans.
-610	500: Internal Server Error	Unexpected Error while updating exercise plan.
-611	404: Not Found	id user not found for searched exercise plan.
-612	500: Internal Server Error	Unexpected Error while getting user's exercise plans.
-613	500: Internal Server Error	Unexpected Error while deleting exercise plan.
-614	500: Internal Server Error	we can't find this exercise plan to delete.
-615	500: Internal Server Error	failed to insert plan into db.
-900	500: Internal Server Error	Unexpected Error
-901	404: Not Found	Unknown resource requested
-902	405: Method Not Allowed	Unsupported operation for requested URI
-903	400: Bad Request	Output media type not specified. Accept request header missing
-904	406: Not Acceptable	Unsupported output media type. Resources are represented only in application/json
-905	400: Bad Request	Input media type not specified. Content-Type request header missing
-906	500: Internal Server Error	Unable to serve the REST request

Table 9: REST Error Code

5.5 REST API Details

User

- URL: `http://localhost:8080/cycleK-1.0.0/rest/user/id/*`
- Method: PUT
- URL Parameters: `{id}` - the user id
- Data Parameters (request body):

```
{ "id": 1, "name": "name", "surname": "surname", "birthday": "birthday",  
  "gender": "gender", "username": "username", "password": "password" }
```

- Success Response:

Http Code	Content
200 OK	<pre>{ "user": { "id": 1, "name": "name", "surname": "surname", "birthday": "birthday", "gender": "gender", "username": "username", "password": "password" } }</pre>

- Error Response:

Http Code	When	Content
400	Missing some values in the body	<pre>{ "message": { "message": "Bad Request", "errorCode": "-107", "errorDetails": "Cannot update the user, body must contains all parameters.", "error": true } }</pre>
500	Unexpected DB error.	<pre>{ "message": { "message": "Internal Error", "errorCode": "-108", "errorDetails": "Internal Server Error", "Unexpected Error while updating user.", "error": true } }</pre>

Stats

- URL: `http://localhost:8080/cycleK-1.0.0/rest/stats/user/{id}`
- Method: GET
- URL Parameters: {id} - the user id
- Data Parameters: No Data Parameters Needed
- Success Response:

Http Code	Content
200 OK	<pre>{ "resource-list": [{ "bodyStats": { "id": 5, "idUser": 1, "weight": 78.0, "height": 175. 0, "fatty": 13.6, "lean": 23.5, "statsDate": "202 4-04-09"} }]}</pre>

- Error Response:

Http Code	When	Content
404	User do not have any body measure registered	<pre>{ "message": { "message": "Not Found", "errorCode": "-215", "errorDetails": "No stat found for searched user.", " error": true} }</pre>
500	Unexpected DB error.	<pre>{ "message": { "message": "Internal Error", " errorCode": "-214", "errorDetails ": "Unexpected Error while retrieving user stats.", "error": true} }</pre>

Social Network

Description: Delete a comment

- URL: `http://localhost:8080/cycleK-1.0.0/post/comment/{commentId}`
- Method: DELETE
- URL Parameters: {id} - the comment id
- Data Parameters: No Data Parameters Needed
- Success Response:

Http Code	Content
200 OK	<pre>{"comment":{"commentId":1,"postId":1,"userId":1,"commentText":"Great job!"}}</pre>

- Error Response:

Http Code	When	Content
404	Comment to delete not found.	<pre>{"message":{"message":"Not Found","errorCode":"-324","errorDetails":"Comment not found","error":true}}</pre>
500	Unexpected DB error.	<pre>{"message":{"message":"Internal Error","errorCode":"-327","errorDetails":"Unexpected Error while deleting comment.","error":true}}</pre>
405	Unsupported operation. Id of comment to delete not specified.	<pre>{"message":{"message":"Unsupported operation","errorCode":"-902","errorDetails":"Unsupported operation for requested URI","error":true}}</pre>

Diet

Description: SaveDiet

- URL: `http://localhost:8080/cycleK-1.0.0/rest/diet`
- Method: POST
- URL Parameters: No parameters are required in the URL
- Data Parameters (request body) :

```
{ "idUser": 1, "planName": "High Carbs", "diet": {  
  "Monday": { "Breakfast": { "Scrambled eggs": 150, "Spinach": 50, "Whole  
    grain toast": 50 }, "Lunch": { "Quinoa": 100, "Mixed vegetables": { "  
      bell peppers": 50, "cucumber": 50, "tomato": 50 }, "Grilled chicken":  
      150 }, "Dinner": { "Salmon": 150, "Broccoli": 100, "Brown rice": 100 }  
    },  
  "Tuesday": { "Breakfast": { "Greek yogurt": 150, "Strawberries": 100, "  
    Granola": 30 }, "Lunch": { "Turkey": 100, "Avocado": 50, "Lettuce": 50  
    , "Tomato": 50, "Carrot sticks": 100 }, "Dinner": { "Kidney beans": 10  
    0, "Corn": 100, "Bell peppers": 50, "Whole grain garlic bread": 50 } }  
}
```

- Success Response:

Http Code	Content
200 OK	No content

- Error Response:

Http Code	When	Content
400	Missing value idUser in the body	<pre>{ "message": { "message": "Bad Request", "errorCode ": "-107", "errorDetails": "No idUser JSON object found in the request.", "error": true } }</pre>
500	Unexpected DB error.	<pre>{ "message": { "message": "Internal Error", " errorCode": "-214", "errorDetails ": "Unexpected error while saving diet.", "error": true } }</pre>

Meal

Description: List meals for a user

- URL: http://localhost:8080/cycleK-1.0.0/rest/meal/idUser/*
- Method: GET
- URL Parameters: {idUser} - the user id
- Data Parameters (request body): No Data Parameters Nee
- Success Response:

Http Code	Content
200 OK	<pre>{"resource-list": [{"meal": {"id": 1, "idUser": 1, "date": "2024-10-04", "mealType": 3, "meal": [{"idFood": 1, "qty": 80}, {"idFood": 2, "qty": 100}, {"idFood": 3, "qty": 300}]}}]}</pre>

- Error Response:

Http Code	When	Content
404	Missing id in database	<pre>{"message": {"message": "Not Found", "errorCode": "-506", "errorDetails": "idUser not found for searched meals", "error": true}}</pre>
500	Unexpected DB error.	<pre>{"message": {"message": "Internal Server Error", "errorCode": "-507", "errorDetails": "Unexpected Error while retrieving meals", "error": true}}</pre>

Exercise

Description: AddExercisePlan

- URL: `http://localhost:8080/cycleK-1.0.0/rest/exercise_plan`
- Method: POST
- URL Parameters: No parameters are required in the URL
- Data Parameters (request body) :

```
{ "idUser": 1, "planName": 1,
  "plan": [
    { "exercise_id": 1, "weight": 20, "repetition": 12 },
    { "exercise_id": 2, "weight": 25, "repetition": 10 } ] }
```

- Success Response:

Http Code	Content
200 OK	No content

- Error Response:

Http Code	When	Content
400	Some exercise plan values is missing in the body	<pre>{ "message": { "message": "Bad Requenst", " errorCode": "-604", "errorDetails ": "No JSON object found in the request", "error": true } }</pre>
500	When nothing is inserted into the database	<pre>{ "message": { "message": "Internal Error", " errorCode": "-615", "errorDetails ": "failed to insert plan into db ", "error": true } }</pre>

500	Unexpected DB error.	<pre>{ "message": { "message": "Internal Error", " "errorCode": "-605", "errorDetails ": "Unexpected Error while adding exercise plan", "error": true} }</pre>
-----	----------------------	--

6 Group Members Contribution

Kimia Abedini Implementation of all features related to the exercise part.

- DAOs: GetExerciseDao.java, ListAllExercisesDAO.java, AddExercisePlanDAO.java, DeleteExercisePlanDAO.java, GetExercisePlanByUserIdDAO.java, GetExercisePlanDAO.java, ListExercisePlanDAO.java, UpdateExercisePlanDAO.java
- Resources: Exercise.java, ExercisePlan.java
- Rest: GetExerciseByIdRR.java, ListExerciseRR.java, AddExercisePlanRR.java, DeleteExercisePlanRR.java, GetExercisePlanByUserIdRR.java, GetExercisePlanByIdRR.java, ListExercisePlanRR.java, UpdateExercisePlanRR.java
- Service: ExerciseService.java, ExercisePlanService.java

Martina Boscolo Bacheto Implementation of all the features related to Posts, Likes and Comments tables:

- DAOs: CreatePostDAO.java, DeletePostDAO.java, GetPostDAO.java, ListPostByUserIdDAO.java, ListPostDAO.java, LoadPostPhotoDAO.java, UpdatePostDAO.java, CountCommentByPostIdDAO.java, CreateCommentsDAO.java, DeleteCommentDAO.java, ListCommentsByPostIdDAO.java, CountLikesByPostIdDAO.java, CreateLikeDAO.java, DeleteLikeDAO.java, ListLikesByPostIdDAO.java
- Resources: Post.java, Like.java, Comment.java
- Rest: CountLikeRR.java, CreateLikeRR.java, DeleteLikeRR.java, ListLikeRR.java, CountCommentRR.java, CreateCommentRR.java, DeleteCommentRR.java, ListCommentRR.java, CreatePostRR.java, DeletePostRR.java, GetPostRR.java, ListPostByUserRR.java, ListPostRR.java, UpdatePostRR.java.
- Service: PostService.java
- Servlets: CreatePostServlet.java, LoadPostPhotoServlet.java
- Jsp pages: create-post-form.jsp, create-post-result.jsp

Alessio Cocco Initial contribution on the project structure setup. Implementation of common functions classes and interfaces (DAO, Dispatcher, Resources).

- ER Schema.
- DAOs: bodyObjective and bodyStats packages.
- Resources: UserGoals.java, UserStats.java, Imc.java, FoodStats.java and MealAdapter.java.
- Rest: userGoals and userStats packages.
- Service: UserGoalService.java and UserStatsService.java.

Giacomo Munerotto Initial contribution on the project structure setup; Implementation of all the features related to Dietplans table:

- Class Diagram.
- Sequence Diagram.
- DAOs: GetDietDAO.java, SaveDietDAO.java, UpdateDietDAO.java
- Resources: Diet.java
- Rest: ListDietRR.java, GetDietByIdRR.java, GetDietByUserIdRR.java, SaveDietRR.java, UpdateDietRR.java
- Service: DietService.java

Marco Tomaiuoli Initial contribution on the project structure setup. Implementation of the all the features related to Users table:

- DAOs: GetUserDAO.java, LoginUserDAO.java, RegisterUserDAO, UpdateUserDAO.java
- Resources: User.java
- Rest: GetUserByIdRR.java, ListUserRR.java, LoginUserRR.java, RegisterUserRR.java, UpdateUserRR.java
- Service: UserService.java
- Utils: PassWordHashing.java, TokenJWT.java

Riccardo Trevisiol Implementation of all features related to the Meal part.

- DAOs: GetFoodDao.java, RegisterFoodDAO.java, GetMealDao.java, RegisterMealDao.java
- Resources: Food.java, Meal.java
- Rest: GetFoodByIdRR.java, ListFoodRR.java, RegisterFoodRR.java, GetMealByUserIdRR.java, ListMealRR.java, RegisterMealRR.java
- Service: FoodService.java, MealService.java

Note: During the drafting of the report, each member contributed to their respective section in accordance with the specified code divisions listed above.