Design

Design for ScholariZer

Dominik Jentsch[1], *Tim Wolk*[2],
Alexander Möhring[3], *Pablo Schmeiser*[4],
Adham Gouda[5]

[1]Phase-leader for Definition of Requirements
[2]Phase-leader for Design
[3]Phase-leader for Implementation
[4]Phase-leader for Quality Assurance
[5]Phase-leader for Final Presentation

Contents

1 High-level Architecture

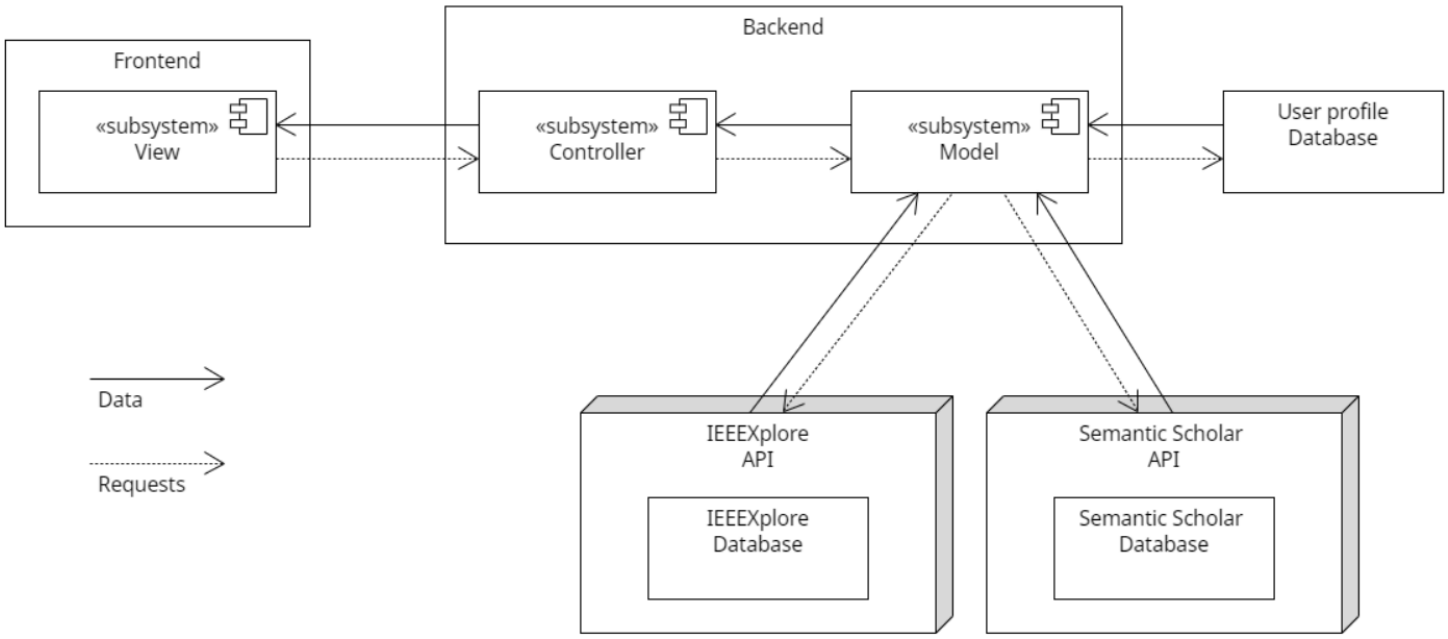


Figure 1: Architecture sketch

ScholariZers architecture resembles the *Model-View-Controller Architecture-pattern* (MVC) with an additional data layer.
MVC is utilized for its clear separation of presentation logic, flow control and business logic, as well as its natural encapsulation of data. It also allows the different components to be reused or modified separately without the need to adjust or rework the other components.

The front end consists of the view subsystem, which will handle all of the presentation logic, i.e. all the interaction between the user and the website as well as the layout of the website.
It will communicate with the back end via HTTP-requests which will be received by the controller subsystem (utilizing the Spring framework) which will then pass all information to the model, where all of the calculations take place.

The model also handles the communication with all the different databases.
All the needed data about authors and papers will be fetched through the Semantic Scholar/IEEEXplore APIs while all information about the users (login data, bookmarks, etc.) will be stored in a separate SQL database, maintained using Spring data JPA.

ScholariZer will also make use of an *opague layered architecture*[6], where the separate MVC subsystems represent the different layers.

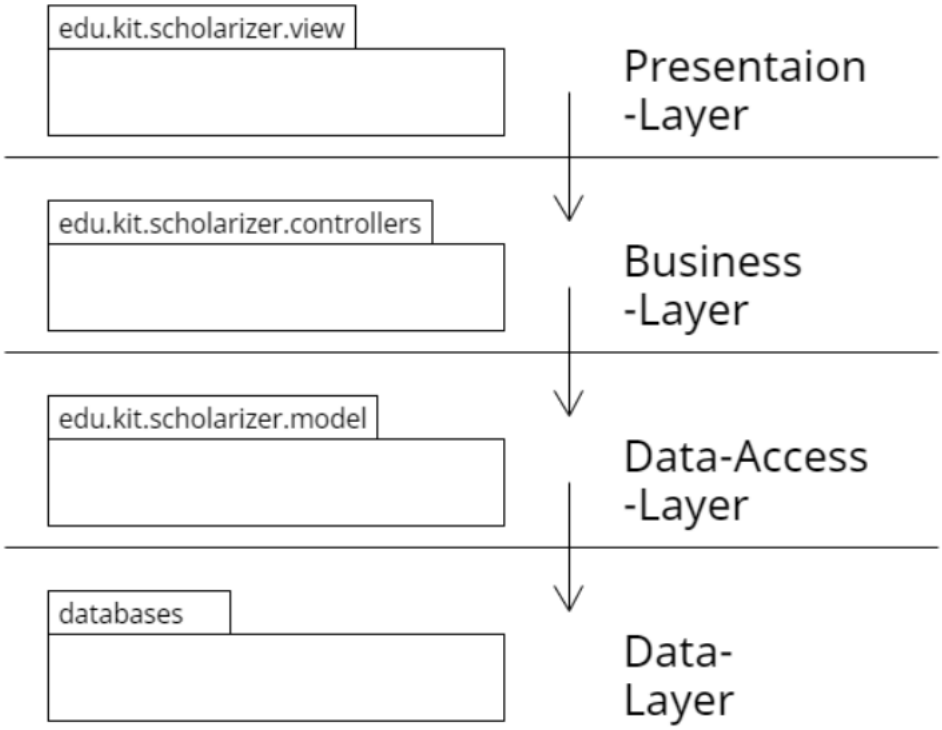


Figure 2: Layered architecture

The dependencies between the MVC components are solely in the direction of the arrows to ensure high maintainability.

[6]Meaning that each subsystem can only access the one right "below" it.

2 Components

2.1 View

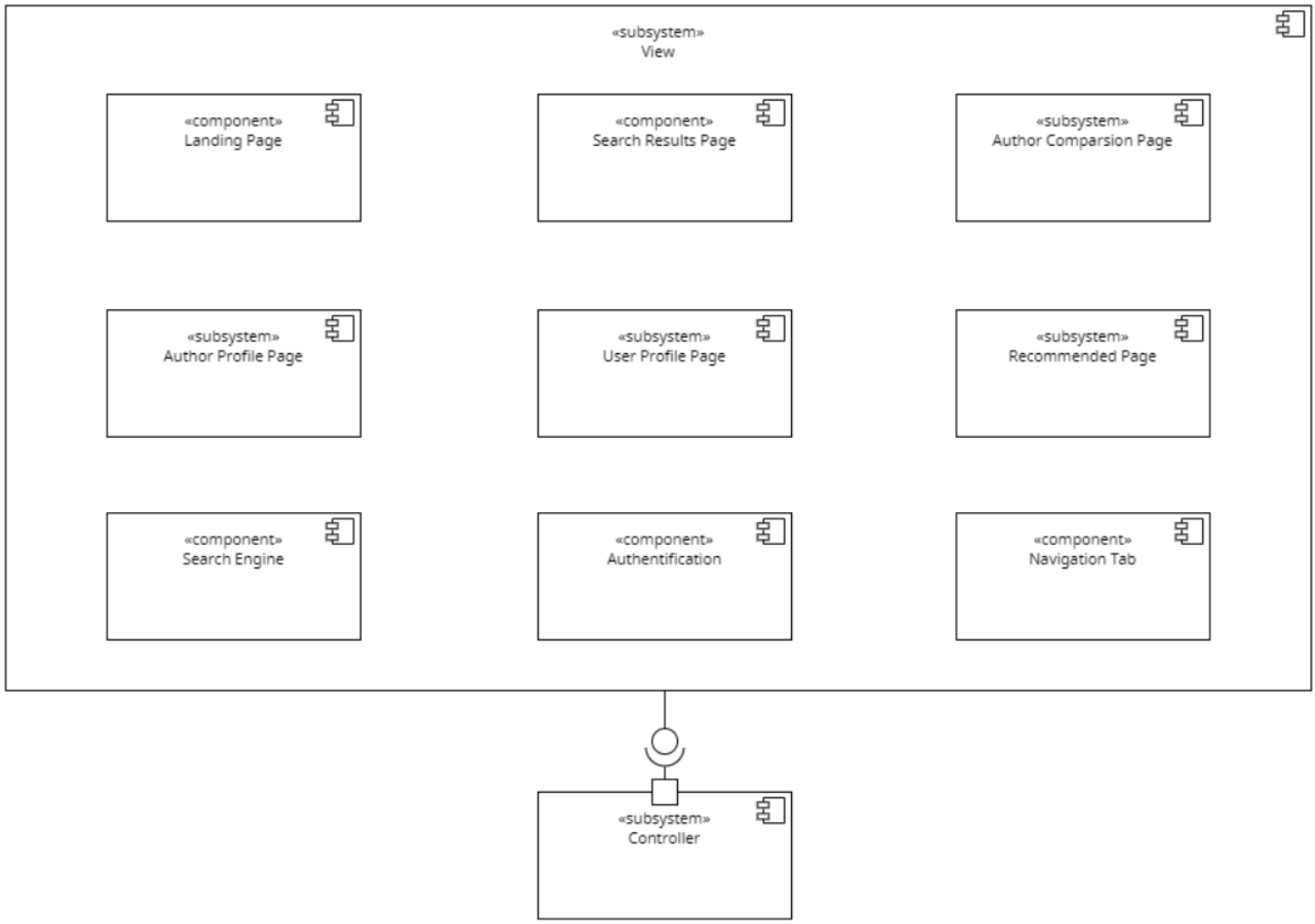


Figure 3: View

The view subsystem will be the part of the application the user directly interacts with. It offers multiple different pages (see below).

Some components, namely the search engine, navigation tab and authentication will be present on all pages across the website.

2.1.1 Landing page

This will be the default page where the user will be directed to when opening the website.

The page will consist of the search bar, the navigation tab and the login/register button.

2.1.2 Search results page

This will be the page that is shown after a search query has been processed.
Here, all search results will be shown separated into authors and papers and the option will be given to change the sorting, show only authors or show only papers.

The page will consist of 20 search entry containers which represent one search result each. On the bottom of the page there will be a button to show the next 20.

2.1.3 Author comparison page

This page will offer the comparison feature where a user can pick up to four different authors and compare them based on multiple different metrics.

The page will consist of multiple "add author" buttons, which will be used to add an author to the comparison. Once an author has been added, the respective button will be replaced by a statistics tab which will show a selection of metrics regarding this author.

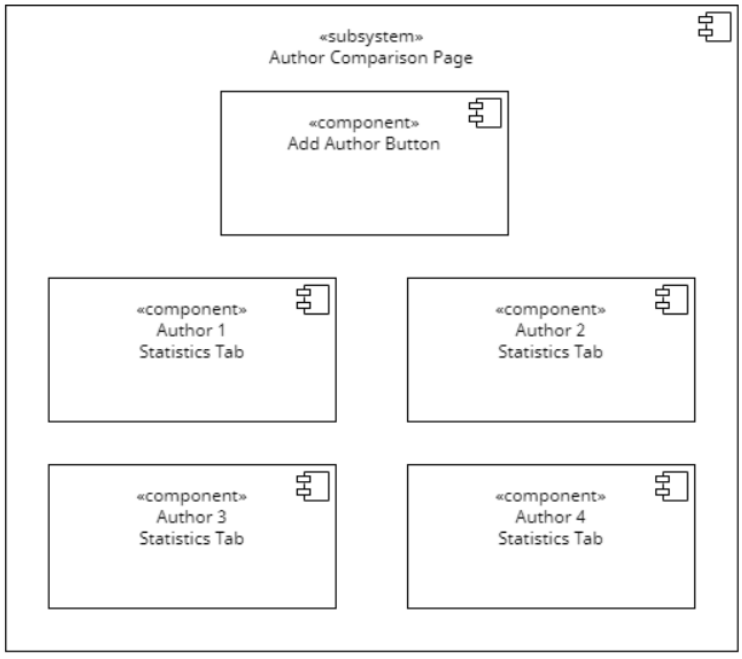


Figure 4: Comparison page

2.1.4 Author profile page

This page shows an authors profile, including all of the available information on him/her.

The page will consist of a banner, showing some basic information about the author, as well a publications page and a statistics page which can be accessed via different buttons. On the profile, there will be a button to switch to the comparison page (see Figure 4) with the author automatically added, as well as a follow button[7]

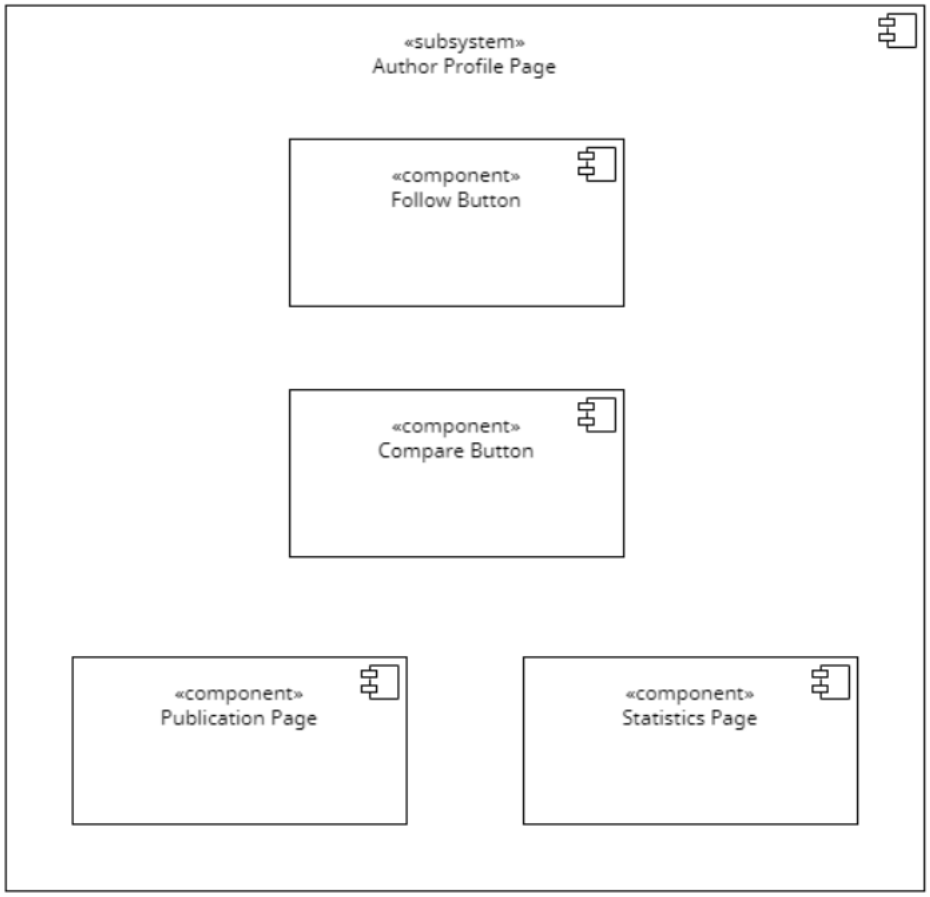


Figure 5: Author profile

[7]Only for logged in users

2.1.5 User profile page

This page will show the users own profile, where he/she can set interests, change his/her e-mail and change his/her password.

The page will consist of an interests tab, where the user can add interests by typing them out, as well as an user info tab, where changes to login data can be made.

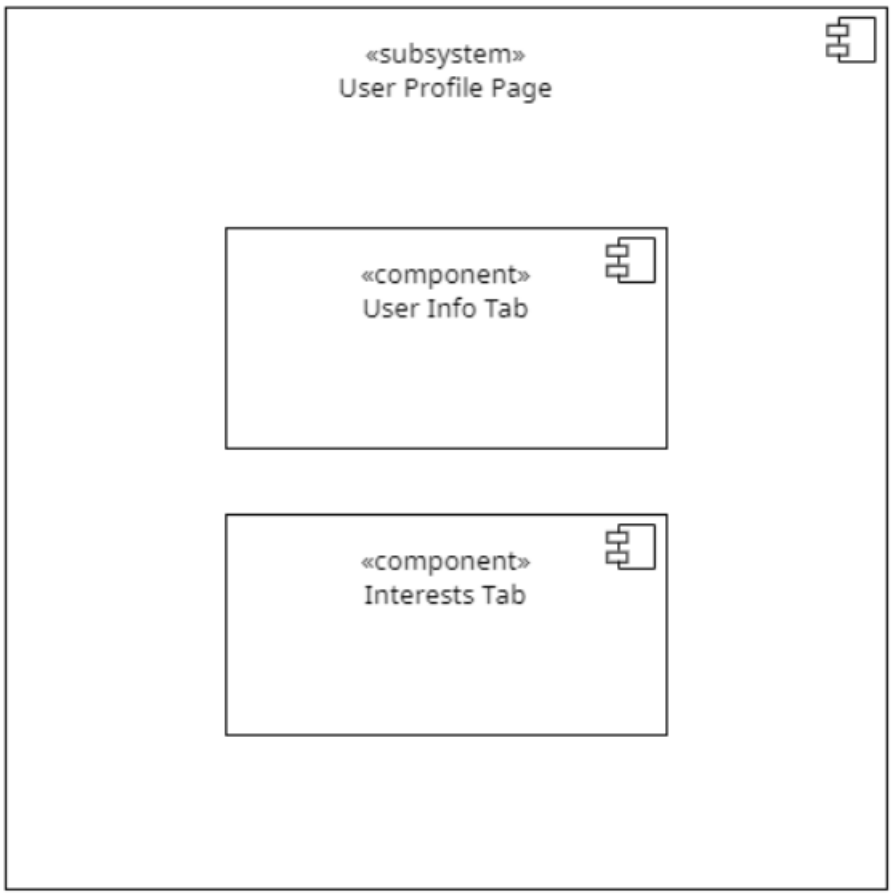


Figure 6: User profile

2.1.6 Recommended page

This page will show the recommendations that are generated based on the users interests and followed authors.

The page will just consist of a news feed, which will be filled with recommended papers and authors, generated from the users interests and follows.

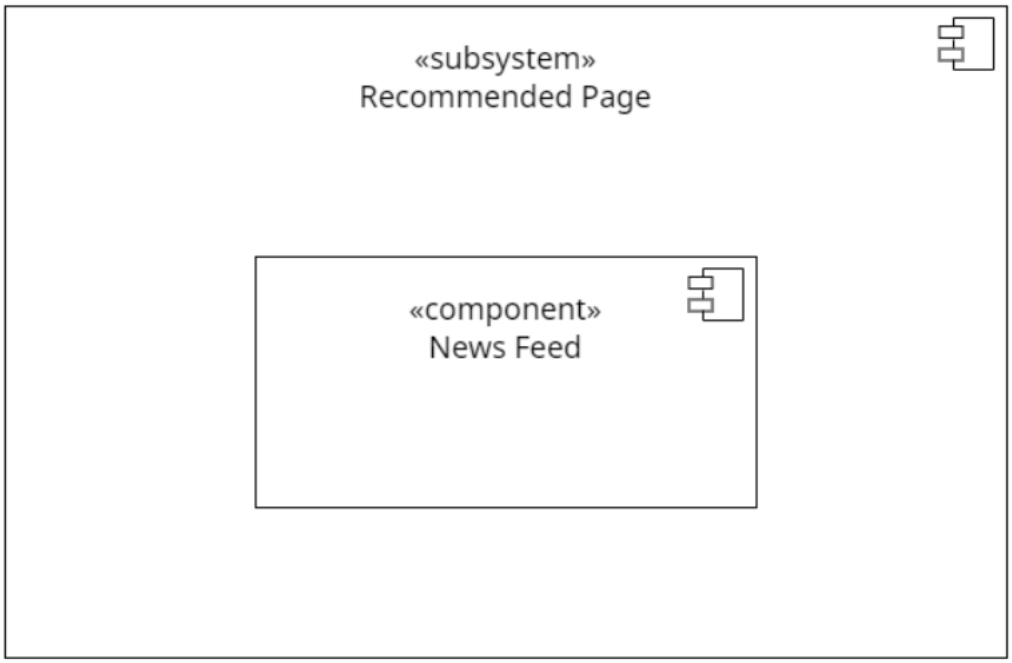


Figure 7: Recommended page

2.1.7 Sign up page

This page will double as sign up/in page and offer the user the option to create an account or to log in into an existing one.

The page will consist of a sign up entry window, which can be over-layed on top any other page and a sign up/in button which is used to sign up/in.

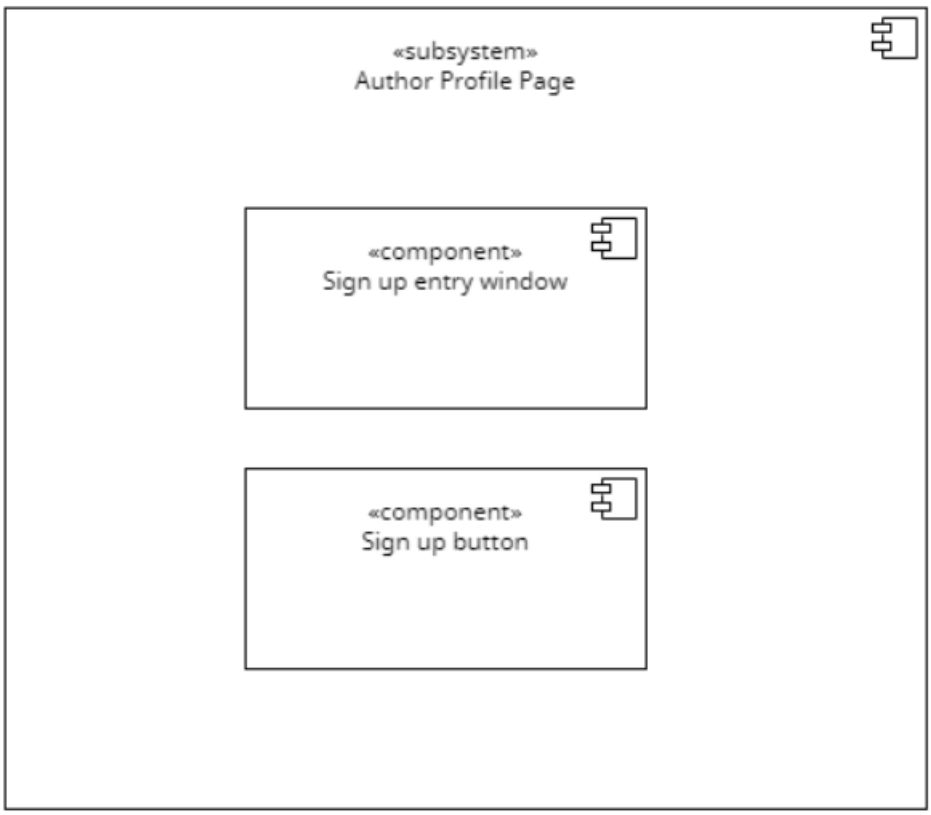


Figure 8: Sign up page

2.2 Controller

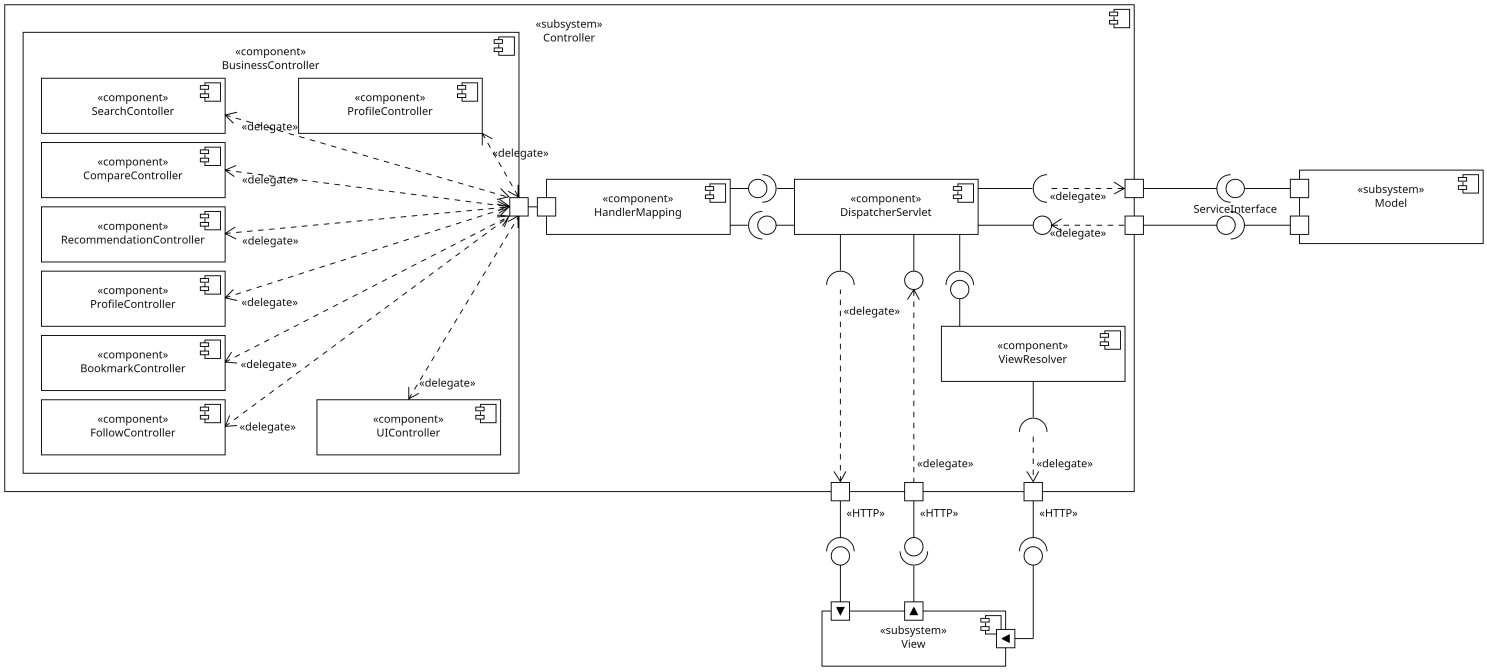


Figure 9: Controller

The controller subsystem will be the part of the system that is responsible for all communication between front end and back end. This is achieved using the Spring framework, which enables the communication between the Java back end and the TypeScript front end via HTTP-requests.

The controller receives the HTTP-requests from the view, using the DispatcherServlet and HandlerMapping components provided by Spring to delegate the requests to the different sub-controllers, which will then call the needed part of the model, so that the needed data can be supplied to the view.

2.3 Model

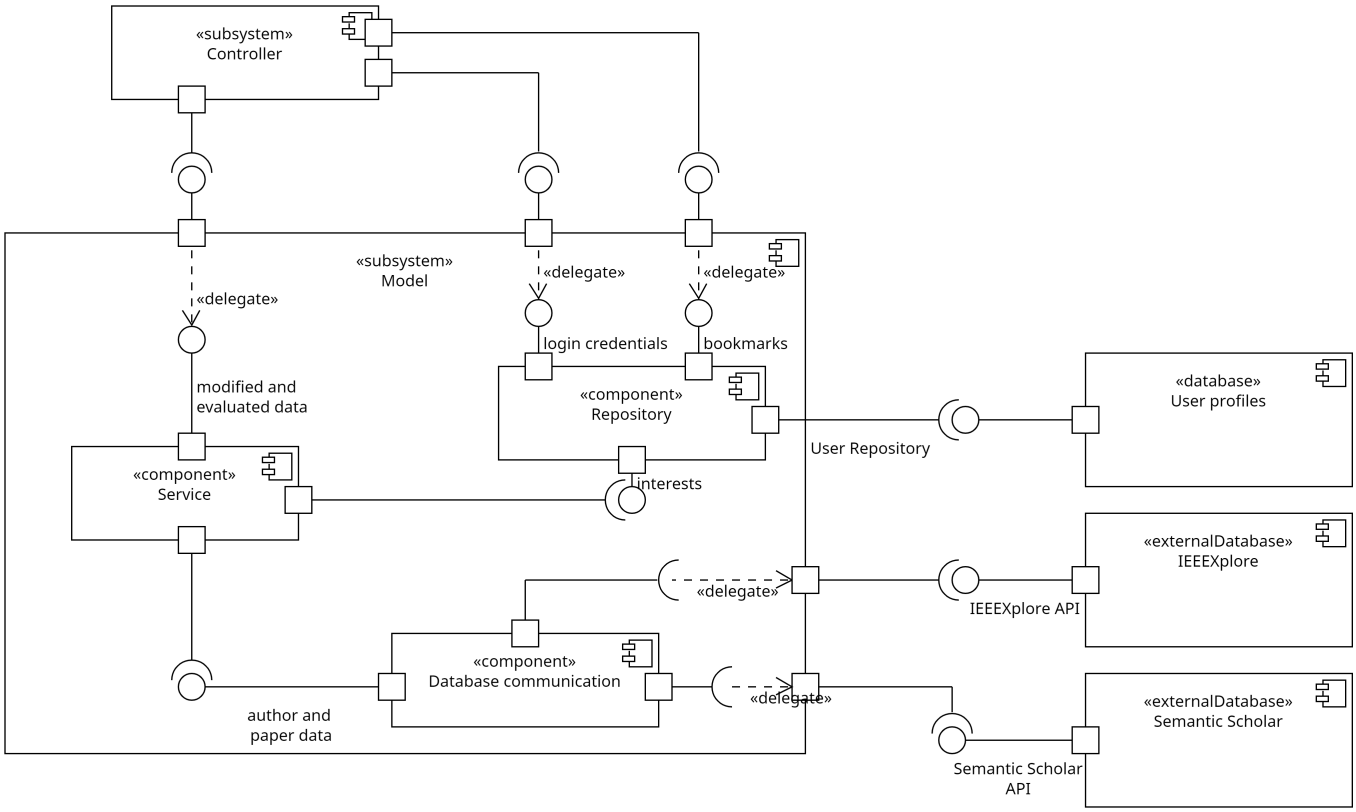


Figure 10: Model

The model subsystem will be responsible for receiving the correctly formatted requests by the controller and fetch/calculate all the different data that is needed.
To achieve this, the model is structured into three components, the repository, the service and the database communication.

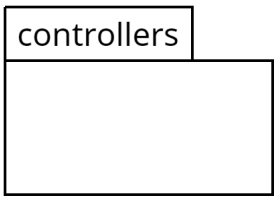
The repository will be responsible for handling all user related data, as well as reading/writing this data from/to the SQL-database storing all user profiles. It will also be responsible for generating session tokens, so that the user is able to log in.

The service will be responsible for all author and paper data related requests, like generating responses from a search query.
It will also calculate all additional data, which can't be directly fetched via an API.

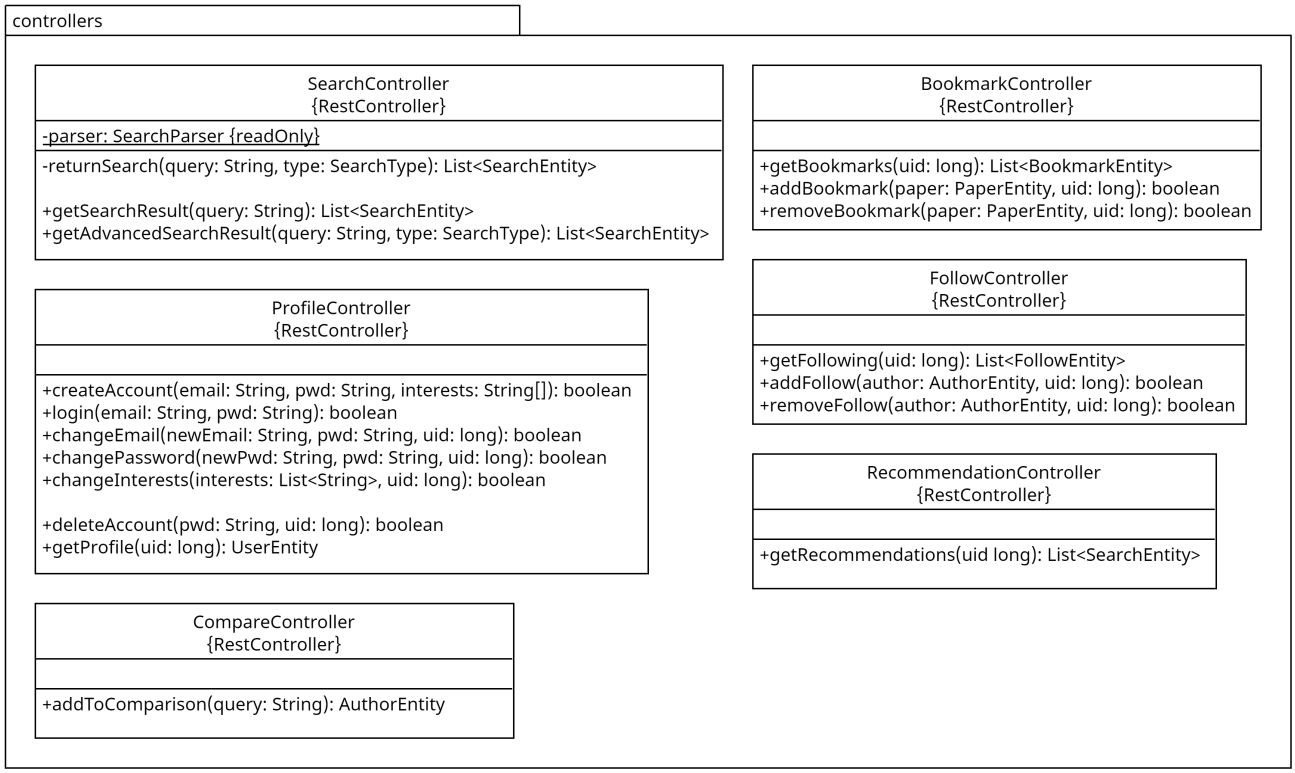
The database communication will be the component that makes the actual API-calls and checks the validity of the responses.

3 Packages and Classes

Controller



3.1 Package controller



3.1.1 Class SearchController

This controller handles incoming HTTP requests to the /search path. It handles these HTTP requests by calling the models' SearchParser, which performs the business logic and afterward returns the models results. These results are then returned by the controller via HTTP to the device that was requesting the information.
It especially handles GET requests to /search/, which are standard search requests and GET requests to /search/filtered, which also should contain a parameter to filter by as well as the search query required by both of these search operations.

Declaration

- @RestController @RequestMapping("/search") class SearchController

Attributes

- private SearchParser parser - The SearchParser used to search the model for the required data. Is used by returnSearch.

Constructors

- public SearchController() - default constructor

Methods

- private List<SearchEntitiy> returnSearch(String query, SearchType type, Comparator sortedBy) - used by getSearch and getAdvancedSearch

- @GetMapping("/") List<SearchEntitiy> getSearchResult(String query) - returns a list of SearchEntities, representing the requested search results to the supplied query to the system requesting it on /search/.
- @GetMapping("/filtered") List<SearchEntitiy> getAdvancedSearchResult(String query, Searchtype type) - returns a list of SearchEntities, representing the requested search results to the supplied query and the supplied type to filter by, to the system requesting it on /search/filtered.

3.1.2 Class CompareController

This controller handles incoming HTTP requests to the /compare path. It handles these HTTP requests by using the SearchController and transforming the received data and afterward returning the results via HTTP to the device that was requesting the information. It especially handles GET requests to /compare/, which are requesting a user to be added to the comparison interface in the frontend, and it requires receiving this searched for author in the request.

Declaration

- @RestController @RequestMapping("/compare") class CompareController

Attributes

- none

Constructors

- public CompareController() - default constructor

Methods

- @GetMapping("/") AuthorEntitiy addToComparison(AuthorEntity author) - returns an AuthorEntitiy, representing the requested author to the supplied query to the system requesting it on /compare/.

3.1.3 Class FollowController

This controller handles incoming HTTP requests to the /follow path. It handles these HTTP requests by calling the models' FollowRepository, which performs the business logic and afterward returns the models results. These results are then returned by the controller via HTTP to the device that was requesting the information.
It especially handles GET requests to /follow/, which are requests about the followed authors of a specified user (identified by his uid), PUT requests to /follow/, which are requests to follow a specified author from a specified user (identified by his uid) and DELETE reuests to /follow/, which are requests to remove the follow of a specified user (identified by his uid) from a specified author.

Declaration

- @RestController @RequestMapping("/follow") class FollowController

Attributes

- none

Constructors

- public FollowController() - default constructor

Methods

- @GetMapping("/") List<FollowEntity> getFollowing(long uid) - returns a list of FollowEntities, representing the requested list of authors, a supplied user follows, to the system requesting it on /follow/.
- @PutMapping("/") boolean addFollow(AuthorEntity author, long uid) - returns a response code, representing the success of the insertion of the supplied author to the follows of the user, to the system requesting it on /follow/.
- @DeleteMapping("/") boolean removeFollow(AuthorEntity author, long uid) - returns a response code, representing the success of the deletion of the supplied author from the follows of the user, to the system requesting it on /follow/.

3.1.4 Class BookmarkController

This controller handles incoming HTTP requests to the /bookmark path. It handles these HTTP requests by calling the models' BookmarkRepository, which performs the business logic and afterward returns the models results. These results are then returned by the controller via HTTP to the device that was requesting the information.
It especially handles GET requests to /bookmark/, which are requests about the bookmarked papers of a specified user (identified by their uid), PUT requests to /bookmark/, which are requests to bookmark a specified paper from a specified user (identified by their uid) and DELETE reuests to /bookmark/, which are requests to remove the bookmark of a specified user (identified by their uid) from a specified paper.

Declaration

- @RestController @RequestMapping("/bookmark") class BookmarkController

Attributes

- none

Constructors

- public BookmarkController() - default constructor

Methods

- @GetMapping("/") List<BookmarkEntity> getBookmarks(long uid) - returns a list of BookmarkEntities, representing the requested list of bookmarks, a supplied user has bookmarked, to the system requesting it on /bookmark/.
- @PutMapping("/") boolean addBookmark(PaperEntity paper, long uid) - returns a response code, representing the success of the insertion of the supplied bookmark to the bookmarks of the user, to the system requesting it on /bookmark/.
- @DeleteMapping("/") boolean removeBookmark(PaperEntity paper, long uid): boolean - returns a response code, representing the success of the deletion of the supplied bookmark from the bookmarks of the user, to the system requesting it on /bookmark/.

3.1.5 Class RecommendationController

This controller handles incoming HTTP requests to the /recommended path. It handles these HTTP requests by calling the models' RecommendationHandler, which performs the business logic and afterward returns the models results. These results are then returned by the controller via HTTP to the device that was requesting the information. It especially handles GET requests to /recommended/, which are requests to retrieve the recommended content for a specified user (identified by their uid).

Declaration

- @RestController @RequestMapping("/recommended") class RecommendationController

Attributes

- none

Constructors

- public RecommendationController() - default constructor

Methods

- @GetMapping("/") List<SearchEntity> getRecommendations(long uid) - returns a list of SearchEntities, representing the requested list of recommendations, a supplied user are recommended to, to the system requesting it on /recommended/.

3.1.6 Class ProfileController

This controller handles multiple different incoming HTTP requests concerning changes or access to the user database. It handles these HTTP requests by calling the models' UserRepository, which performs the business logic and afterward returns the models results. These results are then returned by the controller via HTTP to the device that was requesting the information.
It especially handles POST requests to /register, which are requests to create a new user account using an email, a password and a list of interests, POST requests to /login, which are requests to log a user in using their email address and password, POST requests to /change-email, which are requests to change a users email address. To do this, the new email, the user's password and the user (identified by their uid) are required. Further it handles POST requests to /change-password for which the new password, the old password, as well as the user (identified by their uid) are required. This controller also handles PUT requests to /change-interests, which require a list of interests as well as the user (identified by their uid), it also handles DELETE requests to /delete, which require the user's password and the user (identified by their uid). At last the ProfileController also handles GET requests to /user requesting data on the user's profile.

Declaration

- @RestController class ProfileController

Attributes

- none

Constructors

- public ProfileController() - default constructor

Methods

• @PostMapping("/register") boolean createAccount(String email, String pwd, List<String> interests) - returns a response code, representing the success of the creation of a new user account, using the supplied user's email, password, as well as their list of interests, to the system requesting it on /register.
• @PostMapping("/login") boolean login(String email, String pwd) - returns a response code, representing the success of the login of the user identified by their supplied email and confirming it with their supplied password, to the system requesting it on /login.
• @PostMapping("/change-email") boolean changeEmail(String newEmail, String pwd, long uid) - returns a response code, representing the success of the insertion of the supplied new email, confirming the supplied user's password, to the supplied users email, to the system requesting it on /change-email.
• @PostMapping("/change-password") boolean changePassword(String newPwd, String pwd, long uid) - returns a response code, representing the success of the insertion of the supplied new password, confirming the supplied user's current password, to the supplied user's password, to the system requesting it on /change-password.
• @PutMapping("/change-interests") boolean changeInterests(List<String> interests, long uid) - returns a response code, representing the success of the insertion of the supplied interests, confirming the supplied user's password, to the supplied users interests, to the system requesting it on /change-interests.
• @DeleteMapping("/delete") boolean deleteAccount(String pwd, long uid) - returns a response code, representing the success of the deletion of the user's account, confirming the supplied user's password, to the system requesting it on /delete.
• @GetMapping("/user") UserEntity getProfile(long uid) - returns a UserEntity, representing the supplied user's profile, to the system requesting it.

Model

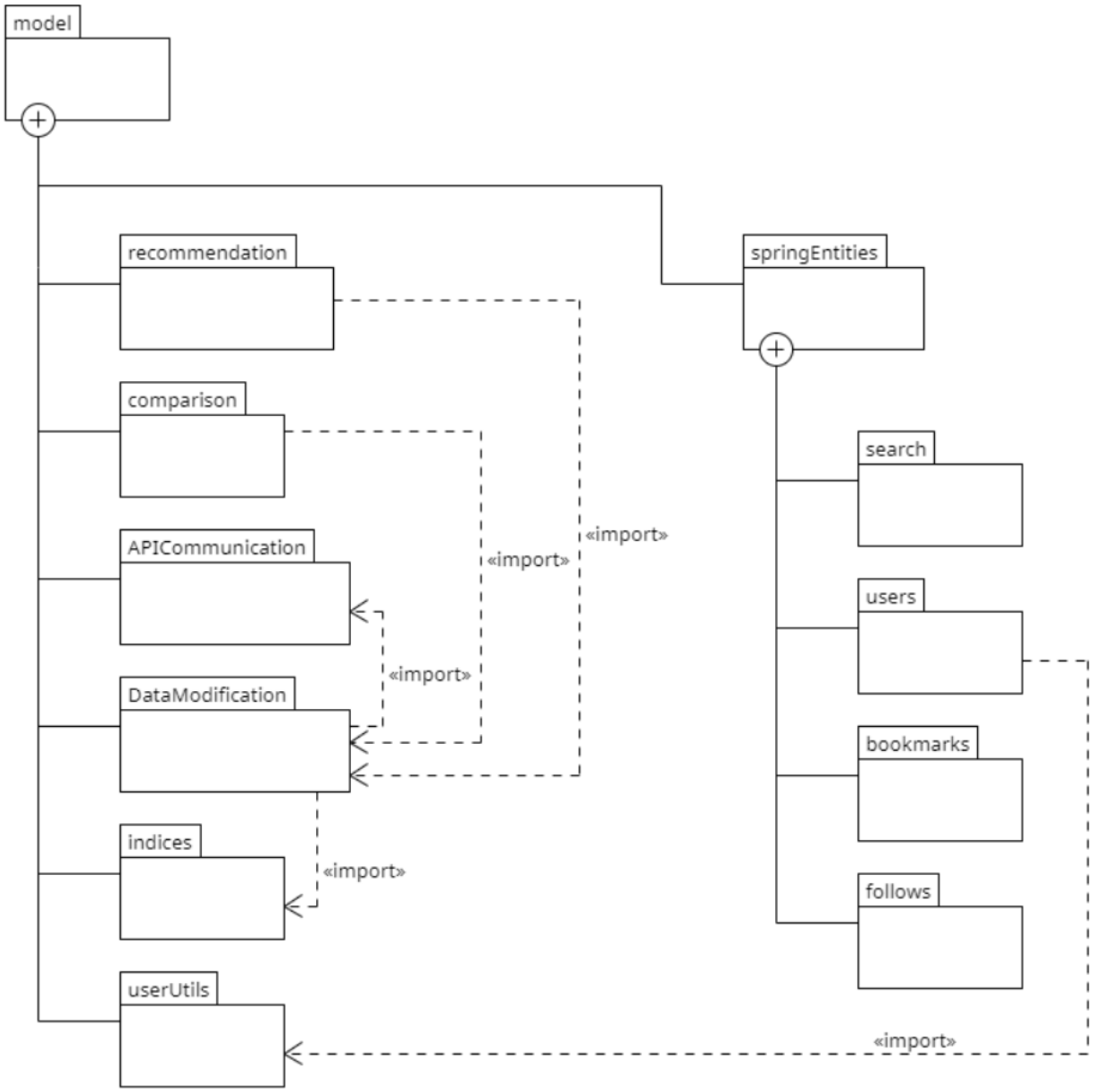


Figure 11: Model package diagram

3.2 Package apiCommunication

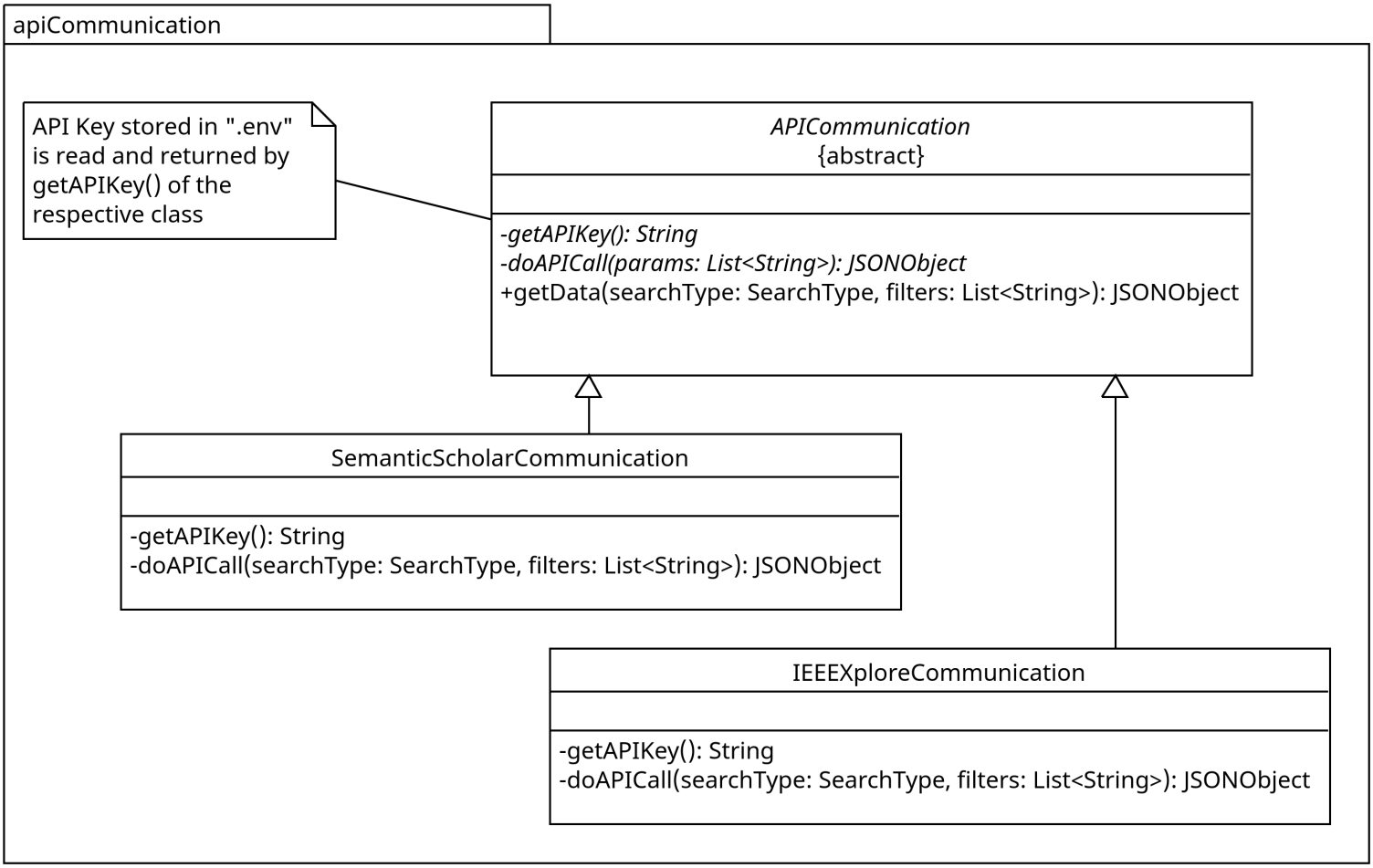


Figure 12: Package apiCommuication

3.2.1 Abstract APICommunication

Template class for working with the external APIs using the Strategy pattern.

Declaration

- abstract APICommunication

Attributes

- none

Constructors

- public APICommunication() - default constructor

Methods

- private abstract String getAPIKey()
- private abstract JSONObject doAPICall(List<String> params)
- public JSONObject getData(SearchType searchType, List<String> filters)

3.2.2 Class SemanticScholarCommunication

Implements the concrete strategy to communicate with the semantic scholar API.

Declaration

- class SemanticScholarCommunication
- extends APICommunication

Attributes

- none

Constructors

- public SemanticScholarCommunication() - default constructor

Methods

- private String getAPIKey()
- private JSONObject doAPICall(List<String> params)

3.2.3 Class IEEEXploreCommunication

Implements the concrete strategy to communicate with the IEEEXplore API.

Declaration

- class IEEEXploreCommunication
- extends APICommunication

Attributes

- none

Constructors

- public IEEEXploreCommunication() - default constructor

Methods

- private String getAPIKey()
- private JSONObject doAPICall(List<String> params)

3.3 Package indices

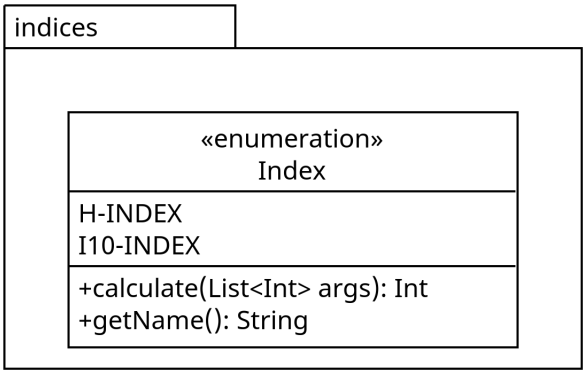


Figure 13: Package indices

3.3.1 Enum Index

Enumeration that holds all indices that need to be calculated, as well as the means to calculate them, resembling a command pattern.

Declaration

- enum Index

Enumerations

- H-INDEX
- I10-INDEX

Attributes

- private String name

Constructors

- private Index(String name)

Methods

- public int calculate(List<Integer> args)
- public String getName()

3.4 Package dataModification

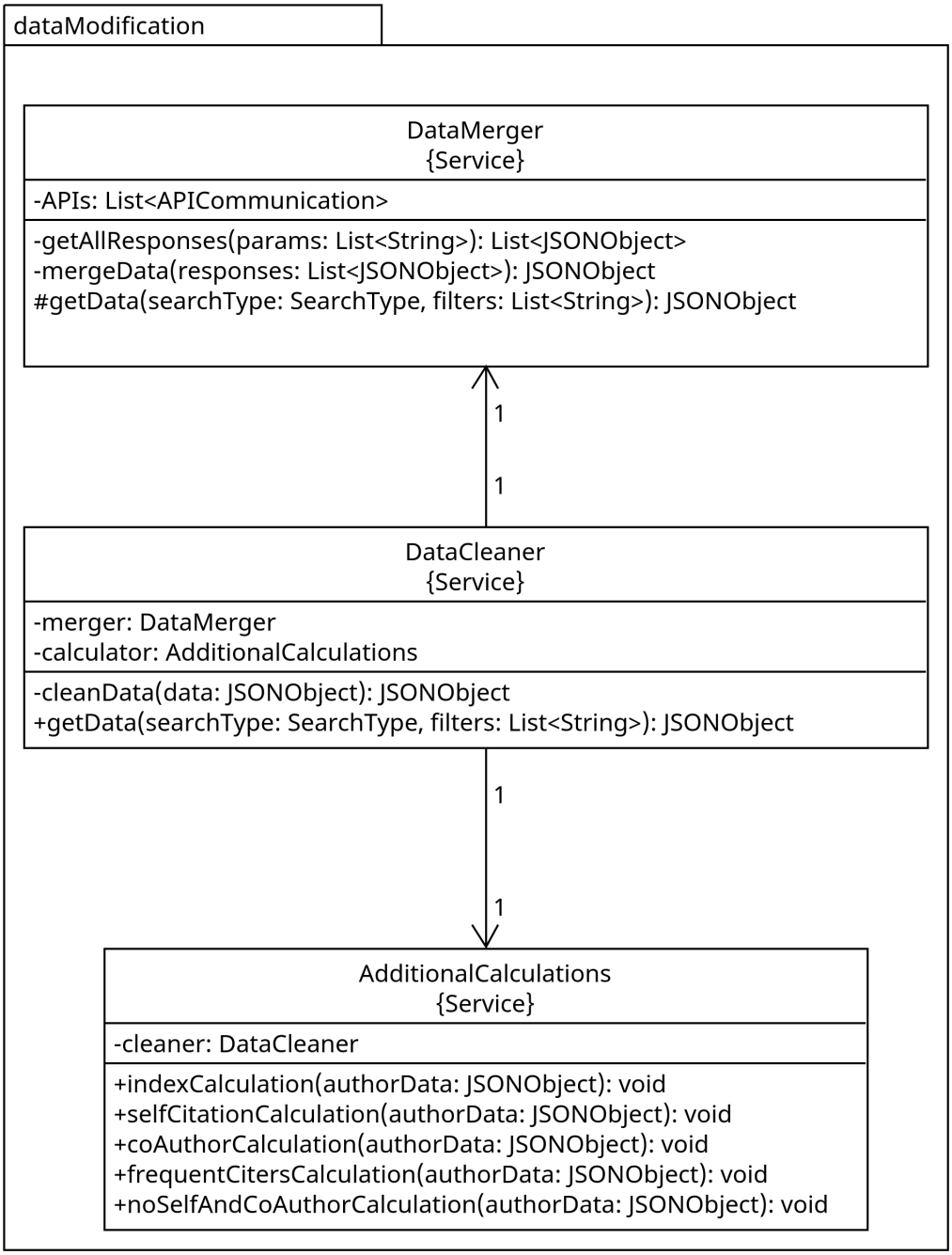


Figure 14: Package dataModification

3.4.1 Class DataMerger

Delegates the request for data to the individual API communication classes and merges all valid responses into a single JSON object.

Declaration

- @Service class DataMerger

Attributes

- private List<APICommunication> apis

Constructors

- public DataMerger(List<APICommunication> apis)

Methods

- private List<JSONObject> getAllResponses(List<String> params)
- private JSONObject merge(List<JSONObject> responses)
- protected JSONObject getData(SearchType searchType, List<String> filters)

3.4.2 Class AdditionalCalculations

Provides the possibilities to calculate multiple different metrics based on the API responses and writes those into the JSON response.
This includes:

- indexCalculation: calculates all Indices based on the Index enum
- selfCitationCalculation: calculates the amount of self citations
- coAuthorCalculation: calculates the amount of citations by frequent co authors
- frequentCitersCalculation: determines the authors that frequently cite papers by the author which information gets accessed currently
- noSelfAndNoCoAuthorCalculation: calculates the amount of citations without frequent co authors and self citations

Declaration

- @Service class AdditionalCalculations

Attributes

- none

Constructors

- public AdditionalCalculations() - default constructor

Methods

- public void indexCalculation(JSONObject authorData)
- public void selfCitationCalculation(JSONObject authorData)
- public void coAuthorCalculation(JSONObject authorData)
- public void frequentCitersCalculation(JSONObject authorData)
- public void noSelfAndCoAuthorCalculation(JSONObject authorData)
- public String generateReference(JSONObject paperData)

3.4.3 Class DataCleaner

The data cleaner class resembles a facade for the dataModification package.
It gets called from outside of the package, delegates the request to the dataMerger class, removes all unnecessary or duplicate data from the JSON response and calls the different calculation methods of the additionalCalculations class.

Declaration

- @Service class DataCleaner

Attributes

- private DataMerger merger
- private AdditionalCalculations calculator

Constructors

- public DataCleaner(DataMerger merger, AdditionalCalculations calculator)

Methods

- private JSONObject cleanData(JSONObject data)
- public JSONObject getData(SearchType searchType, List<String> filters)

3.5 Package comparison

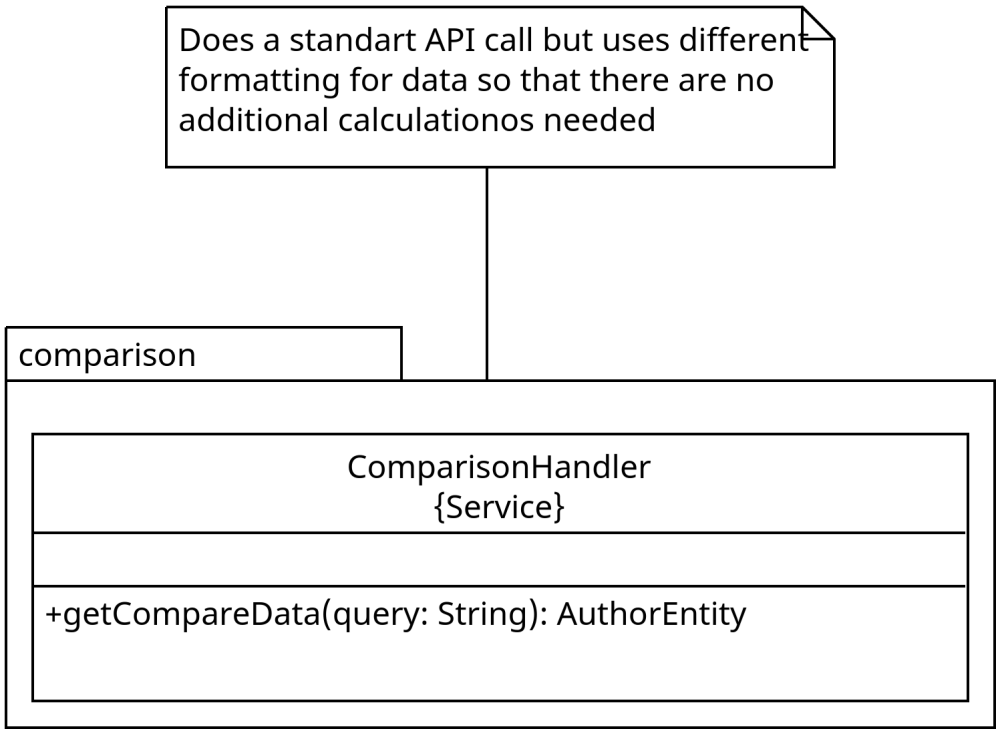


Figure 15: Package comparison

3.5.1 Class ComparisonHandler

Starts the API call process and Entity creation process for the author comparison feature.

Declaration

- @Service class ComparisonHandler

Attributes

- none

Constructors

- public ComparisonHandler() - default constructor

Methods

- public AuthorEntity getCompareData(String query)

3.6 Package recommendation

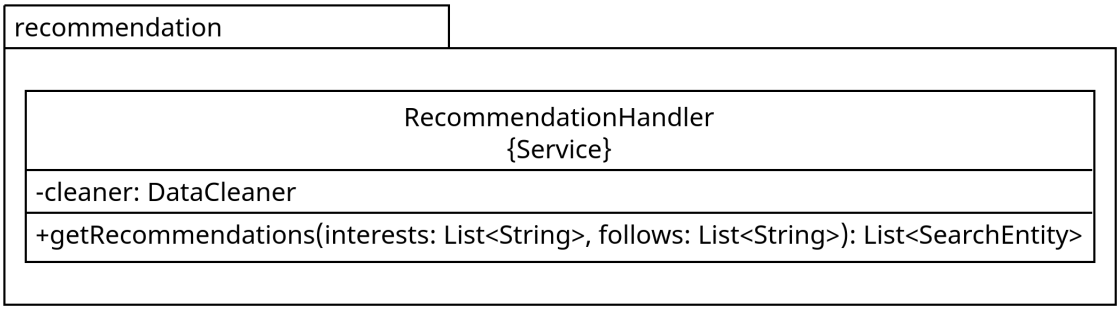


Figure 16: Package recommendation

3.6.1 Class RecommendationHandler

Accesses a users interests and follows on login and initiates API calls to generate recommendations for said user.

Declaration

- @Service class RecommendationHandler

Attributes

- private DataCleaner cleaner

Constructors

- public RecommendationHandler(DataCleaner cleaner)

Methods

- public List<SearchEntity> getRecommendations(List<String> interests, List<String> follows)

3.7 Package userUtils

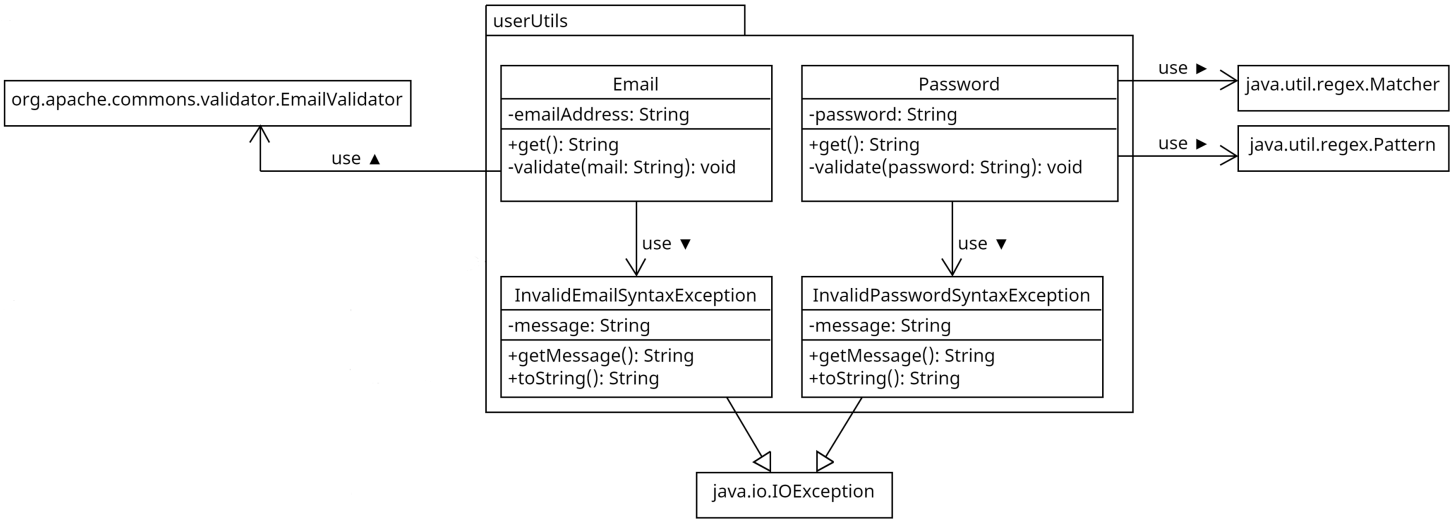


Figure 17: Package userUtils

3.7.1 Class InvalidEmailSyntaxException

Exception thrown in case user tries to register with an email which syntax is invalid.

Declaration

- class InvalidEmailSyntaxException
- extends java.io.IOException

Attributes

- private String message

Constructors

- public InvalidEmailSyntaxException()
- public InvalidEmailSyntaxException(String message)

Methods

- public String getMessage()
- public String toString()

3.7.2 Class InvalidPasswordSyntaxException

Exception thrown in case user tries to register with a password which syntax is invalid.

Declaration

- class InvalidPasswordSyntaxException
- extends java.io.IOException

Attributes

- private String message

Constructors

- public InvalidPasswordSyntaxException()
- public InvalidPasswordSyntaxException(String message)

Methods

- public String getMessage()
- public String toString()

3.7.3 Class Email

Represents an email address, which can be validated using the validate method.

Declaration

- class Email

Attributes

- private String emailAddress

Constructors

- public Email(String emailAddress)

Methods

- public String get()
- private void validate(String mail)

3.7.4 Class Password

Represents a password, which can be validated using the validate method.

Declaration

- class Password

Attributes

- private String password

Constructors

- public Password(String password)

Methods

- public String get()
- private void validate(String password)

3.8 Package springEntities

3.9 Subpackage bookmarks

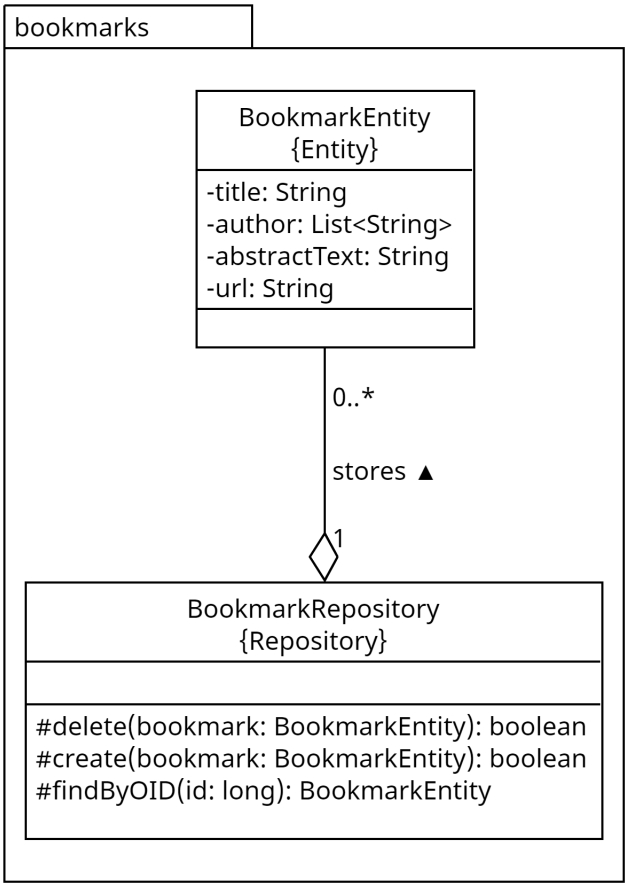


Figure 18: Package bookmarks

3.9.1 Class BookmarkEntity

Represents a database entry for a bookmarked paper.

Declaration

- @Entity class BookmarkEntity

Attributes

- private String title
- private String author
- private String abstract
- private String url

Constructors

- public BookmarkEntity(String title, String author, String abstract, String url)

Methods

- none

3.9.2 Class BookmarkRepository

Handles database access for bookmarked papers.

Declaration

- @Repository class BookmarkRepository

Attributes

- none

Constructors

- public BookmarkRepository() - default constructor

Methods

- protected boolean delete(BookmarkEntity bookmark)
- protected boolean create(BookmarkEntity bookmark)
- protected BookmarkEntity findByOID(long id)

3.10 Subpackage follows

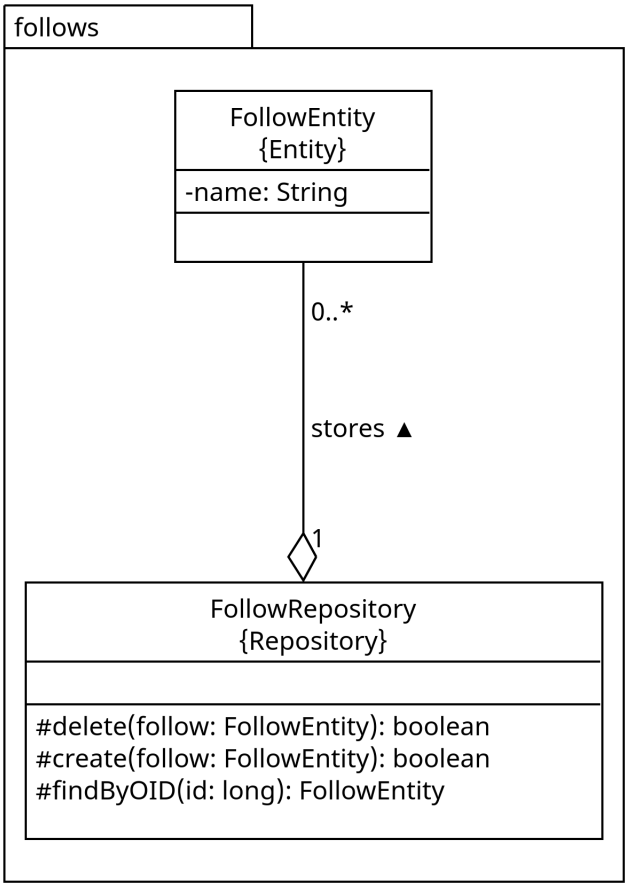


Figure 19: Package follows

3.10.1 Class FollowEntity

Represents a database entry for a followed author.

Declaration

- @Entity class FollowEntity

Attributes

- private String name
- private long id

Constructors

- public FollowEntity(String name, String id)

Methods

- none

3.10.2 Class FollowRepository

Handles database access for follows.

Declaration

- @Repository class FollowRepository

Attributes

- none

Constructors

- public FollowRepository() - default constructor

Methods

- protected boolean delete(FollowEntity bookmark)
- protected boolean create(FollowEntity bookmark)
- protected FollowEntity findByOID(long id)

3.11 Subpackage users

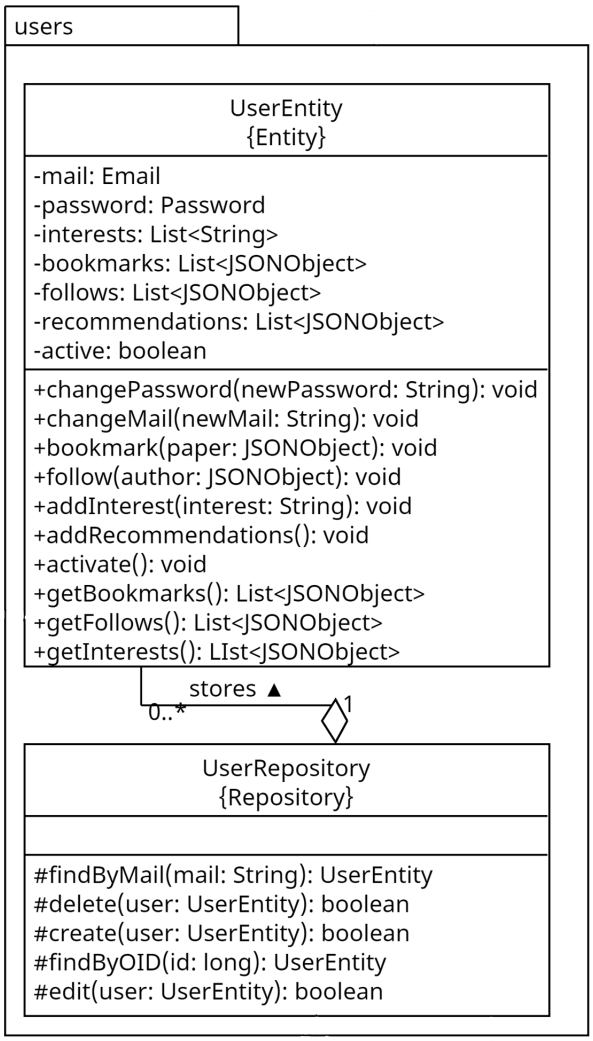


Figure 20: Package users

3.11.1 Class UserEntity

Represents a database entry for a user profile, providing a wide array of options to modify the saved data.

Declaration

- @Entity class UserEntity

Attributes

- private boolean active
- private Email mail
- private Password password
- private List<JSONObject> interests
- private List<JSONObject> bookmarks
- private List<JSONObject> follows
- private List<JSONObject> recommendations

Constructors

- public UserEntity(Email mail, Password password,
List<JSONObject> interests, List<JSONObject> bookmarks,
List<JSONObject> follows, List<JSONObject> recommendations)
- public UserEntity(Email mail, Password password)

Methods

- public void changePassword(String newPassword)
- public void changeMail(String newMail)
- public void bookmark(JSONObjct paper)
- public void follow(JSONObjct auhtor)
- public void addInterest(String interest)
- public void addRecommendations()
- public void activate()
- public List<JSONObject> getBookmarks()
- public List<JSONObject> getFollows()
- public List<JSONObject> getInterests()

3.11.2 Class UserRepository

Handles database access for user profiles.

Declaration

- @Repository class UserRepository

Attributes

- none

Constructors

- public UserRepository() - default constructor

Methods

- protected UserEntity findByMail(String mail)
- protected boolean delete(UserEntity user)
- protected boolean create(UserEntity user)
- protected UserEntity findByOID(long id)
- protected boolean edit(UserEntity user)

3.12 Subpackage search

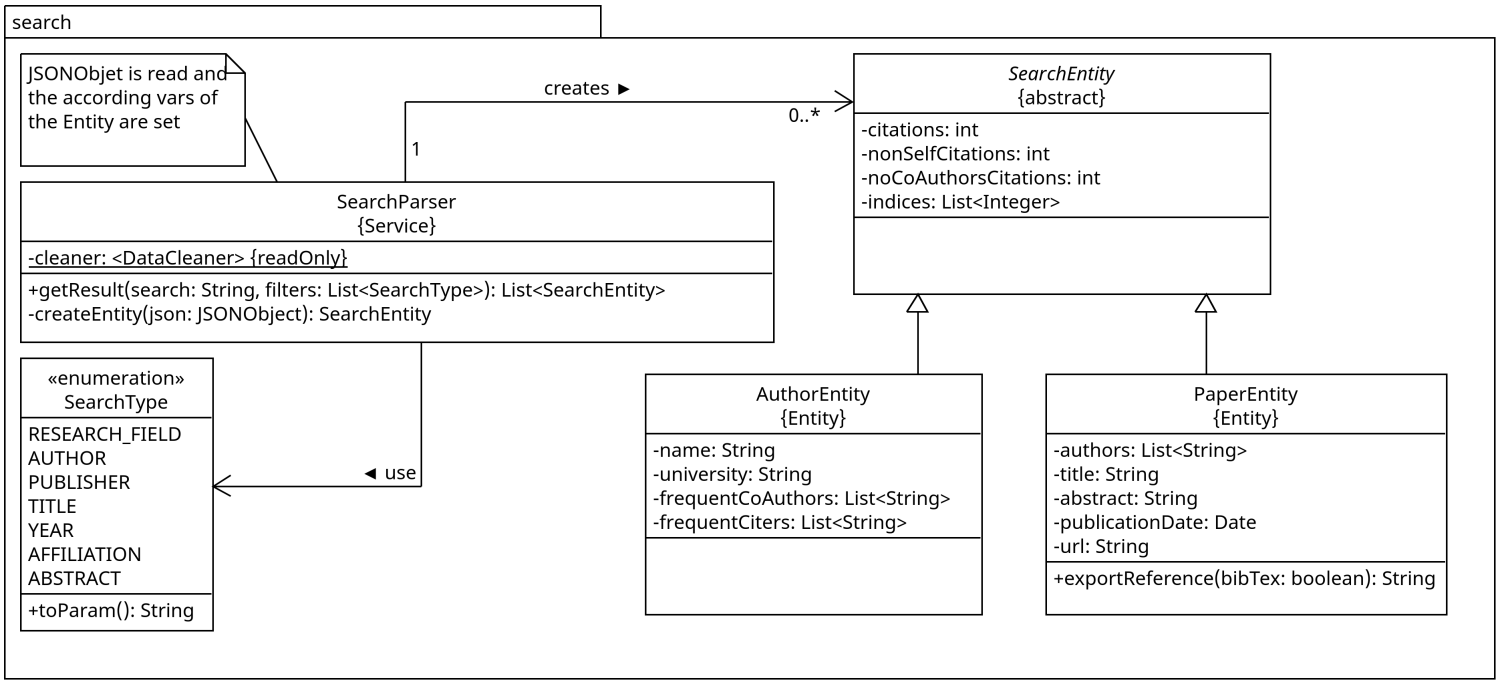


Figure 21: Package search

3.12.1 Abstract SearchEntity

Abstract class to define the basics of a Spring entity representing a search result, which can be converted into JSON format by the according controller.

Declaration

- abstract SearchEntity

Attributes

- private int citations
- private int nonSelfCitations
- private int noCoAuthorCitaitons
- private List<Integer> indices

Constructors

- public SearchEntity(int citaitons, int nonSelfCitations,
int noCoAuthorCitations List<Integer> indices)

Methods

- none

3.12.2 Class AuthorEntity

Defines the specific attributes of an author search result.

Declaration

- @Entity class AuthorEntity
- extends SearchEntity

Attributes

- private String name
- private String university
- private List<String> frequentCoAuthors
- private List<String> frequentCiters

Constructors

- public AuthorEntity(String name, String university, List<String> frequentCoAuthor, List<String> frequentCiters)

Methods

- none

3.12.3 Class PaperEntity

Defines the specific attributes of a paper search result.

Declaration

- @Entity class PaperEntity
- extends SearchEntity

Attributes

- private List<String> authors
- private String title
- private String abstract
- private Date publicationDate
- private String url

Constructors

- public PaperEntity(List<String> authors, String title, String abstract, Data publicationDate, String url)

Methods

- *none*

3.12.4 Enum SearchType

Enumeration of a multitude of search types, which are used to specify the API calls.

Declaration

- enum SearchTyp

Enumerations

- RESEARCH_FIELD
- AUTHOR
- PUBLISHER
- TITLE
- YEAR
- AFFILIATION
- ABSTRACT

Attributes

- private String param

Constructors

- private SearchType(String param)

Methods

- public String toParam()

3.12.5 Class SearchParser

Parses user search input into a search entity and initiates an API call to retrieve the needed data.

Declaration

- @Service class SearchParser

Attributes

- none

Constructors

- public SearchParser() - default constructor

Methods

- public List<SearchEntity> getResult(String search, List<SearchType> filters)
- private SearchEntity createEntity(JSONObject json)

View

3.13 Package App

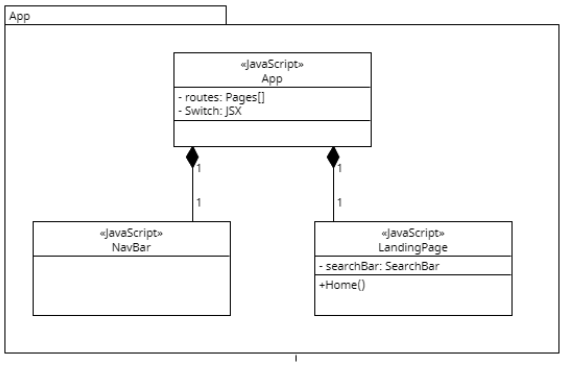


Figure 22: Package App

3.13.1 App

The App is responsible for connecting the sites and the NavBar to a working application. It contains all the sites and the used switches.

Declaration

- App

Props

- private Pages[] routes
- private JSX switch

Methods

- none

3.13.2 NavBar

The NavBar class provides the navigation bar that can be used on any site.

Declaration

- NavBar

Props

- none

Methods

- none

3.13.3 LandingPage

The LandingPage class is responsible for showing the landing page and also uses the SearchBar

Declaration

- LandingPage

Props

- none

Methods

- none

3.14 Package NavLinks

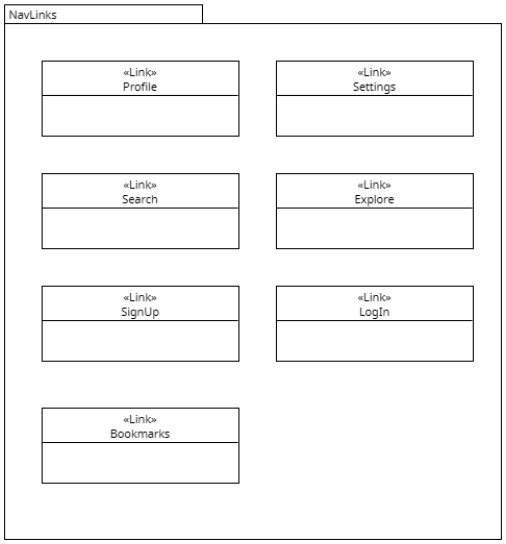


Figure 23: Package NavLinks

3.14.1 Link Profile

The Profile class is responsible for redirecting a user to his profile page by clicking on the button in the NavBar.

Declaration

- Link to = '/profile'

Props

- none

Methods

- none

3.14.2 Link Settings

The Settings class is responsible for redirecting a user to his settings by clicking on the button in the NavBar.

Declaration

- Link to = '/settings'

Props

- none

Methods

- none

3.14.3 Link Search

The Search class is responsible for opening a search bar so that the user can enter the search terms.

Declaration

- Link to = '/search'

Props

- none

Methods

- none

3.14.4 Link Explore

The Explore class is responsible for redirecting a user to his recommended page by clicking on the button in the NavBar.

Declaration

- Link to = '/explore'

Props

- none

Methods

- none

3.14.5 Link SignUp

The Profile class is responsible for redirecting a user to the sign up page by clicking on the button in the NavBar.

Declaration

- Link to = '/sign-up'

Props

- none

Methods

- none

3.14.6 Link LogIn

The LogIn class is responsible for redirecting a user to the log in page by clicking on the button in the NavBar.

Declaration

- Link to = '/sign-up'

Props

- none

Methods

- none

3.14.7 Link Bookmarks

The Bookmarks class is responsible for redirecting a user to the bookmark page by clicking on the button in the NavBar.

Declaration

- Link to = '/sign-up'

Props

- none

Methods

- none

3.15 Package Pages

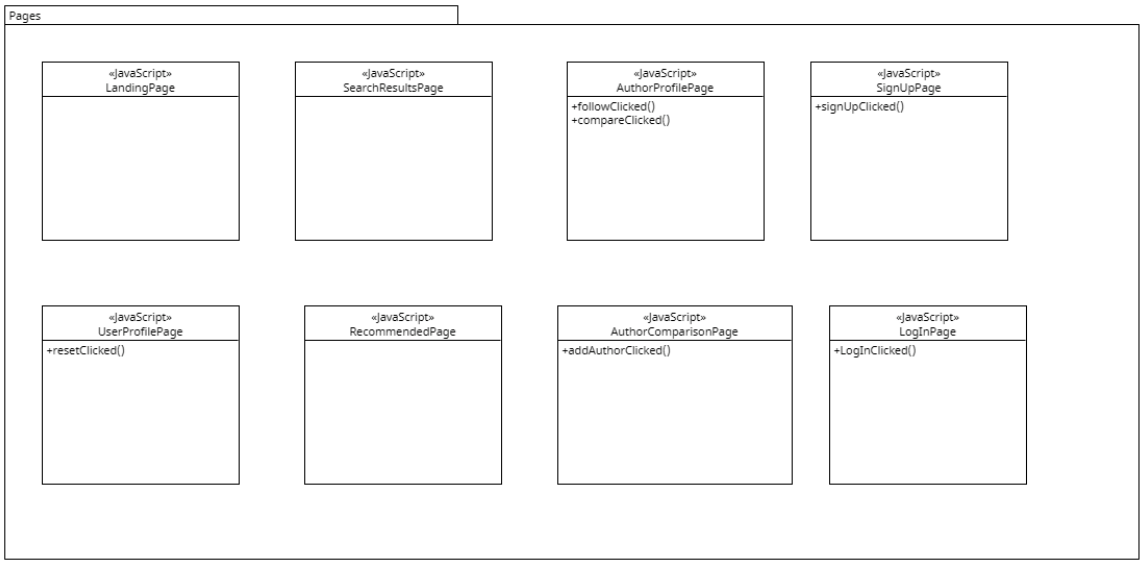


Figure 24: Package Pages

3.15.1 LandingPage

The LandingPage class is responsible for the content that is represented on the landing page and includes a SearchBar.

Declaration

- LandingPage

Props

- none

Methods

- none

3.15.2 SearchResultsPage

SearchResultsPage is responsible for showing all the search results to a user.

Declaration

- SearchResultsPage

Props

- none

Methods

- none

3.15.3 AuthorProfilePage

AuthorProfilePage is responsible for showing the profile of an author.

Declaration

- AuthorProfilePage

Props

- none

Methods

- public void followClicked()
- public void compareClicked()

3.15.4 SignUpPage

SignUpPage is responsible for showing the user a section where he can sign up.

Declaration

- SignUpPage

Props

- none

Methods

- public void signUpClicked()

3.15.5 UserProfilePage

UserProfilePage is responsible for showing the user his own profile.

Declaration

- UserProfilePage

Props

- none

Methods

- public void resetClicked()

3.15.6 RecommendedPage

RecommendedPage is responsible for showing all the recommendations to the user.

Declaration

- RecommendedPage

Props

- none

Methods

- none

3.15.7 AuthorComparisonPage

AuthorComparisonPage is responsible for allowing the user to compare up to 4 authors and show chosen metrics to get an overview.

Declaration

- AuthorComparisonPage

Props

- none

Methods

- public addAuthorClicked()

3.15.8 LogInPage

LogInPage is responsible for displaying the login page.

Declaration

- Link to = '/sign-up'

Props

- none

Methods

- public void loginClicked()

3.16 Package AuthorComparisonPage

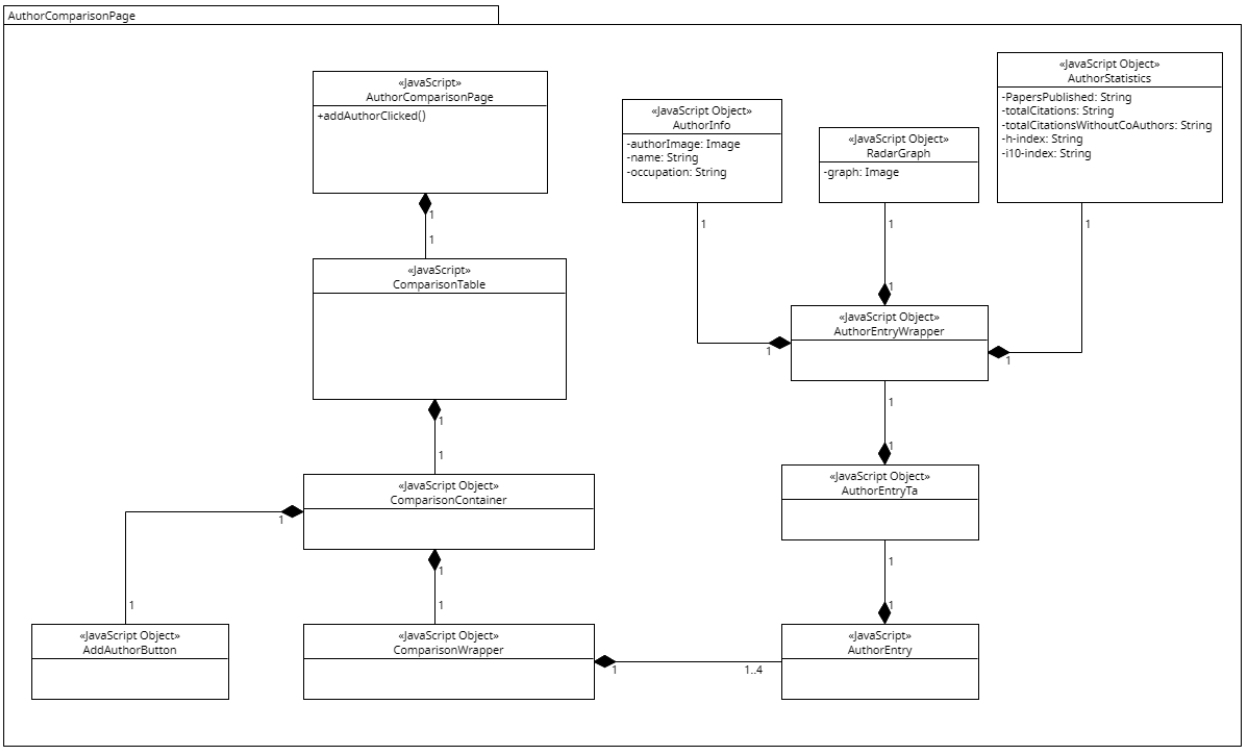


Figure 25: Package AuthorComparisonPage

3.16.1 AuthorComparisonPage

AuthorComparisonPage is responsible for allowing the user to compare up to 4 authors and show chosen metrics to get an overview.

Declaration

- AuthorComparisonPage

Props

- none

Methods

- none

3.16.2 ComparisonTable

ComparisonTablePage is responsible for providing the comparison table for the authors.

Declaration

- ComparisonTable

Props

- none

Methods

- none

3.16.3 ComparisonContainer

ComparisonContainer is providing the containers for the comparison.

Declaration

- ComparisonContainer

Props

- none

Methods

- none

3.16.4 ComparisonWrapper

ComparisonContainer is wrapping the comparison display.

Declaration

- ComparisonWrapper

Props

- none

Methods

- none

3.16.5 AddAuthorButton

AddAuthorButton is providing the button to add an author.

Declaration

- AddAuthorButton

Props

- none

Methods

- none

3.16.6 AuthorEntry

AuthorEntry is providing an author.

Declaration

- AuthorEntry

Props

- none

Methods

- none

3.16.7 AuthorEntryContainer

AuthorEntryContainer is providing the containers for the author.

Declaration

- AuthorEntryContainer

Props

- none

Methods

- none

3.16.8 AuthorEntryWrapper

AuthorEntryWrapper is wrapping all needed information of the author together.

Declaration

- AuthorEntryWrapper

Props

- none

Methods

- none

3.16.9 AuthorInfo

AuthorInfo is providing information of the author.

Declaration

- AuthorImage

Props

- private Image authorImage
- private String name
- private String occupation

Methods

- none

3.16.10 RadarGraph

RadarGraph is providing the radar graph of the author.

Declaration

- RadarGraph

Props

- private Image graph

textitMethods

- none

3.17 Package SearchResultsPage

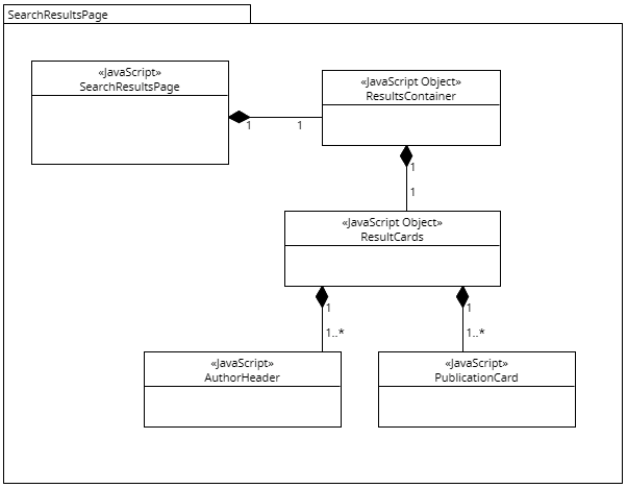


Figure 26: Package SearchResultsPage

3.17.1 SearchResultsPage

SearchResultsPage is responsible for showing the search results to the user.

Declaration

- SearchResultsPage

Props

- none

Methods

- none

3.17.2 ResultsContainer

ResultsContainer is providing a container for all the results.

Declaration

- ResultsContainer

Props

- none

Methods

- none

3.17.3 ResultsCards

ResultsCards is providing the cards.

Declaration

- ResultsCards

Props

- none

Methods

- none

3.17.4 AuthorHeader

AuthorHeader is providing the header of an author.

Declaration

- AuthorHeader

Props

- none

Methods

- none

3.17.5 PublicationResult

PublicationResult is providing the results for the publications found.

Declaration

- PublicationResult

Props

- none

Methods

- none

3.18 Package SignUpPage

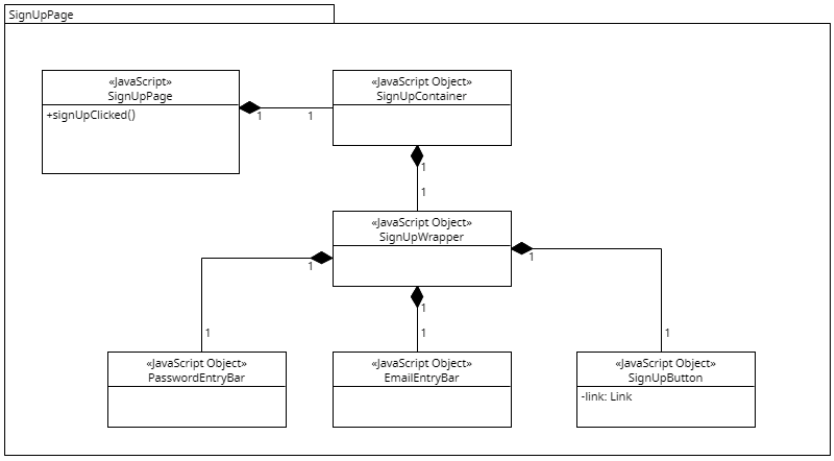


Figure 27: Package SignUpPage

3.18.1 SignUpPage

SignUpPage is providing an interface for the user to sign up.

Declaration

- SignUpPage

Props

- none

Methods

- public void signUpClicked()

3.18.2 SignUpContainer

The SignUpContainer is providing a container for the sign up.

Declaration

- SignUpContainer

Props

- none

Methods

- none

3.18.3 SignUpWrapper

The SignUpWrapper is wrapping the sign up.

Declaration

- SignUpWrapper

Props

- none

Methods

- none

3.18.4 PasswordEntryBar

The PasswordEntryBar is providing a bar for entering the password.

Declaration

- PasswordEntryBar

Props

- none

Methods

- none

3.18.5 EmailEntryBar

The EmailEntryBar is providing a bar for entering the email.

Declaration

- EmailEntryBar

Props

- none

Methods

- none

3.18.6 SignUpButton

The SignUpButton is providing a button to sign up.

Declaration

- SignUpButton

Props

- none

Methods

- none

3.19 Package LandingPage

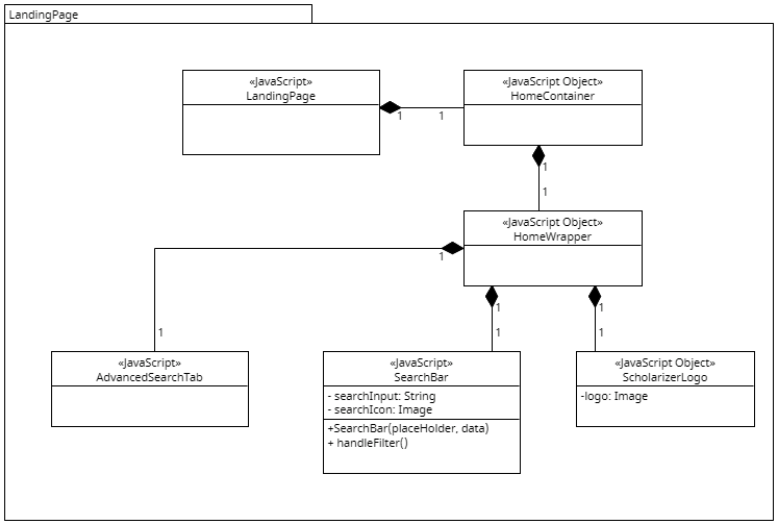


Figure 28: Package LandingPage

3.19.1 LandingPage

The LandingPage is providing the landing page.

Declaration

- LandingPage

Props

- none

Methods

- none

3.19.2 HomeContainer

The HomeContainer is providing a container for the landing page.

Declaration

- HomeContainer

Props

- none

Methods

- none

3.19.3 HomeWrapper

The HomeWrapper is providing a wrapper for the landing page.

Declaration

- HomeWrapper

Props

- none

Methods

- none

3.19.4 AdvancedSearchTab

The AdvancedSearchTab is providing an advanced search tab.

Declaration

- AdvancedSearchTab

Props

- none

textitMethods

- none

3.19.5 SearchBar

The SearchBar is providing a search bar which can also handle filters.

Declaration

- Searchbar

Props

- private String searchInput
- private Image searchIcon

Methods

- public void searchbar(placeholder, data)
- public void handleFilter()

3.19.6 ScholarizerLogo

The ScholarizerLogo is providing the logo of ScholariZer.

Declaration

- ScholarizerLogo

Props

- private Image logo

Methods

- none

3.20 Package UserProfilePage

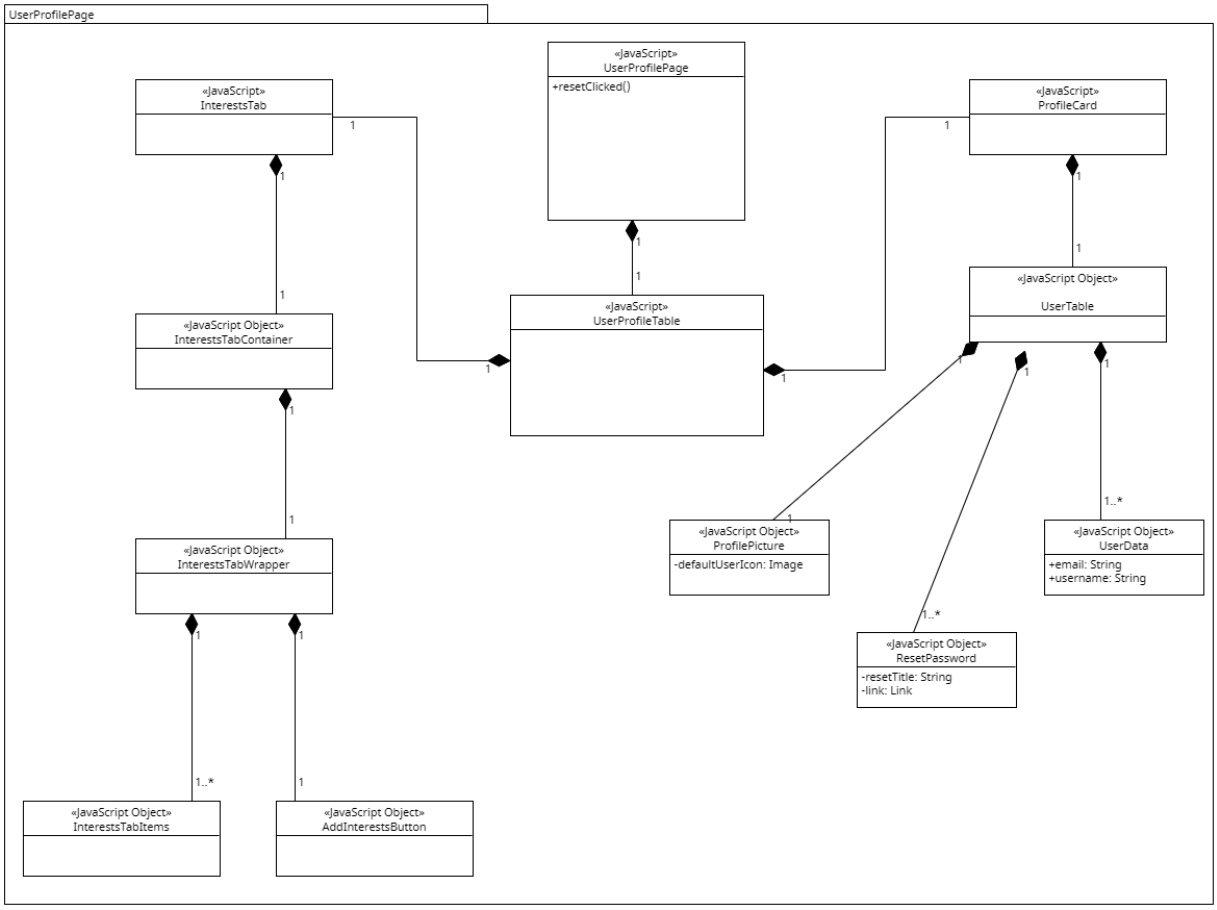


Figure 29: Package UserProfilePage

3.20.1 InterestsTab

The InterestsTab is providing the interests of a user.

Declaration

- InterestsTab

Props

- none

Methods

- none

3.20.2 InterestsTabContainer

The InterestsTabContainer is providing a container for the interests of a user.

Declaration

- InterestsTabContainer

Props

- none

Methods

- none

3.20.3 InterestsTabWrapper

The InterestTabWrapper is wrapping the interests of a user.

Declaration

- InterestsTabWrapper

Props

- none

Methods

- none

3.20.4 InterestsTabItems

The InterestTabItems is providing the interest tabs of a user.

Declaration

- InterestsTabItems

Props

- none

Methods

- none

3.20.5 AddInterestsButton

The AddInterestButton is providing a button to add interests.

Declaration

- AddInterestsButton

Props

- none

textitMethods

- none

3.20.6 UserProfilePage

The UserProfilePage is providing the user profile page of a user.

Declaration

- UserProfilPage

Props

- none

Methods

- public void resetClicked()

3.20.7 ProfileCard

The ProfileCard is providing the profile card of a user.

Declaration

- ProfileCard

Props

- none

Methods

- none

3.20.8 UserTable

The UserTable is providing a table of users.

Declaration

- UserTable

Props

- none

Methods

- none

3.20.9 ProfilePicture

The ProfilePicture is providing the profile picture of a user.

Declaration

- ProfilePicture

Props

- private Image deafultUserIcon

Methods

- none

3.20.10 ResetPassword

The ResetPassword is providing the user the option to reset his password.

Declaration

- ResetPassword

Props

- private String resetTitle
- private Link link

Methods

- none

3.20.11 UserData

The UserData is providing all the data of a user.

Declaration

- UserData

Props

- public String email
- public String username

Methods

- none

3.21 Package RecommendedPage

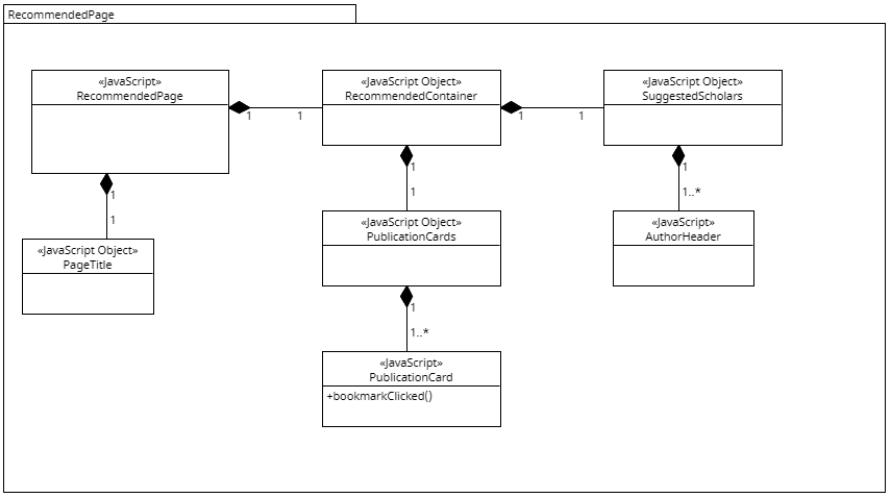


Figure 30: Package RecommendedPage

3.21.1 RecommendedPage

The RecommendedPage is providing the recommended page.

Declaration

- RecommendedPage

Props

- none

Methods

- none

3.21.2 RecommendedContainer

The RecommendedContainer is providing the container for the recommended page.

Declaration

- RecommendedContainer

Props

- none

textitMethods

- none

3.21.3 SuggestestedScholars

The SuggestestedScholars is suggesting scholars.

Declaration

- SuggestestedScholars

Props

- none

Methods

- none

3.21.4 AuthorHeader

The AuthorHeader is providing the header of an author.

Declaration

- AuthorHeader

Props

- none

Methods

- none

3.21.5 PageTitle

The PageTitle is providing the page title.

Declaration

- PageTitle

Props

- none

Methods

- none

3.21.6 PublicationCards

The PublicationCards is providing the card of the publications.

Declaration

- PublicationCards

Props

- none

Methods

- none

3.21.7 PublicationCard

The PublicationCard is providing a card of a publication.

Declaration

- PublicationCard

Props

- none

Methods

- public void bookmarkClicked()

3.21.8 AuthorHeader

The AuthorHeader is providing the header of an author.

Declaration

- AuthorHeader

Props

- none

Methods

- none

3.22 Package AuthorProfilePage

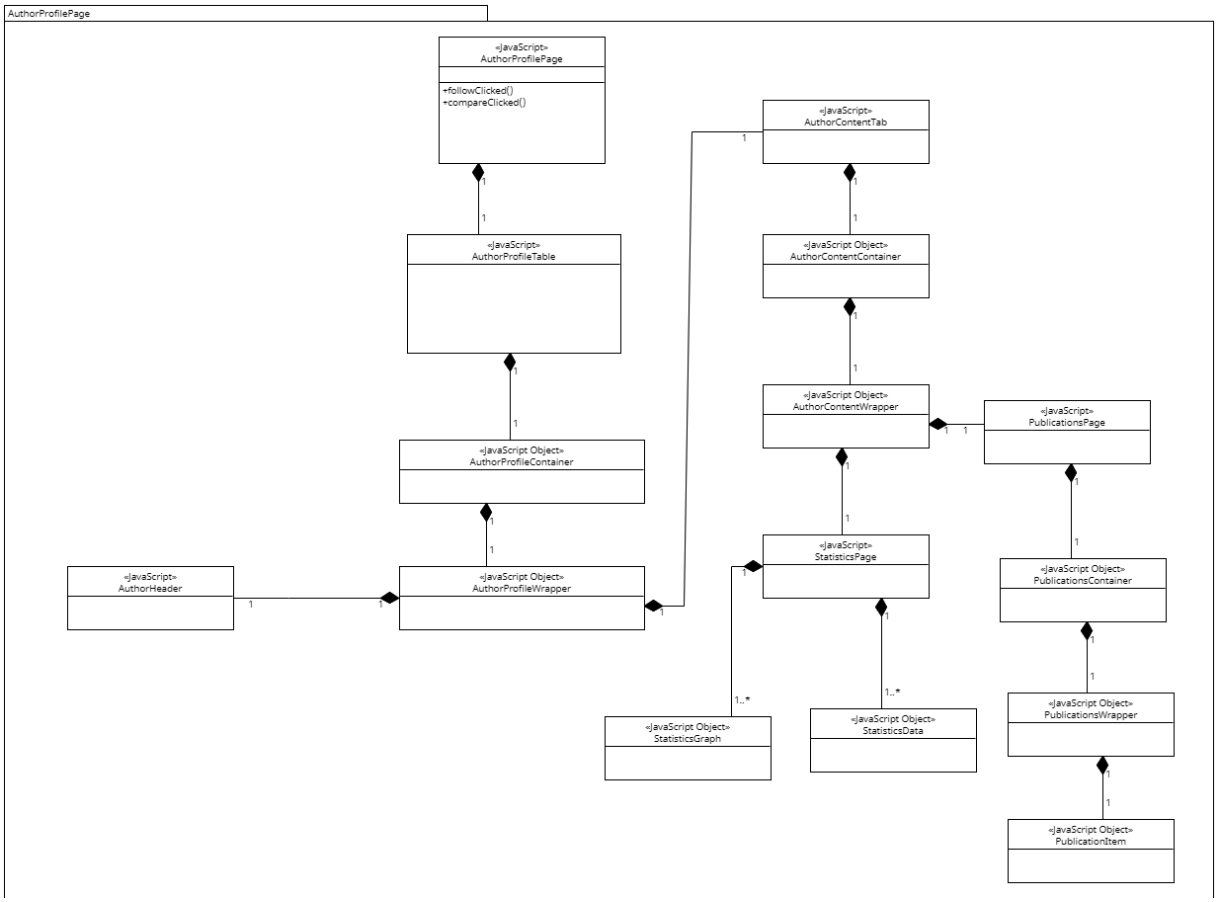


Figure 31: Package AuthorProfilePage

3.22.1 AuthorProfilePage

The AuthorProfilePage is providing the page of an author's profile.

Declaration

- AuthorProfilePage

Props

- none

Methods

- public followClicked()
- public compareClicked()

3.22.2 AuthorProfileTable

The AuthorProfileTable is providing the table of an author's profile.

Declaration

- AuthorProfileTable

Props

- none

Methods

- none

3.22.3 AuthorProfileContainer

The AuthorProfileContainer is providing the container for an author's profile.

Declaration

- AuthorProfileContainer

Props

- none

Methods

- none

3.22.4 AuthorProfileWrapper

The AuthorProfileContainer is wrapping page of an author's profile.

Declaration

- AuthorProfileWrapper

Props

- none

Methods

- none

3.22.5 AuthorHeader

The AuthorHeader is providing the header of an author.

Declaration

- AuthorHeader

Props

- none

Methods

- none

3.22.6 AuthorContentTab

The AuthorContentTab is providing the content of an author.

Declaration

- AuthorContentTab

Props

- none

Methods

- none

3.22.7 AuthorContentContainer

The AuthorContentContainer is providing the container for the content of an author.

Declaration

- AuthorContentContainer

Props

- none

Methods

- none

3.22.8 AuthorContentWrapper

The AuthorContentWrapper is wrapping the content of an author.

Declaration

- AuthorContentWrapper

Props

- none

Methods

- none

3.22.9 PublicationsPage

The PublicationsPage is providing the page for publications.

Declaration

- PublicationsPage

Props

- none

Methods

- none

3.22.10 StatisticsPage

The StatisticsPage is providing the page for the statistics.

Declaration

- StatisticsPage

Props

- none

Methods

- none

3.22.11 PublicationsContainer

The PublicationsContainer is providing the container for publications.

Declaration

- PublicationsContainer

Props

- none

Methods

- none

3.22.12 PublicationsWrapper

The PublicationsWrapper is wrapping the content for publications.

Declaration

- PublicationsWrapper

Props

- none

Methods

- none

3.22.13 PublicationsItem

The PublicationsItem is providing the item as a publication.

Declaration

- PublicationsItem

Props

- none

Methods

- none

3.22.14 StatisticsGraph

The StatisticsGraph is providing the graph for statistics.

Declaration

- StatisticsGraph

Props

- none

Methods

- none

3.22.15 StatisticsData

The StatisticsData is providing the data for the statistics.

Declaration

- StatisticsData

Props

- none

Methods

- none

3.23 Package NavBar

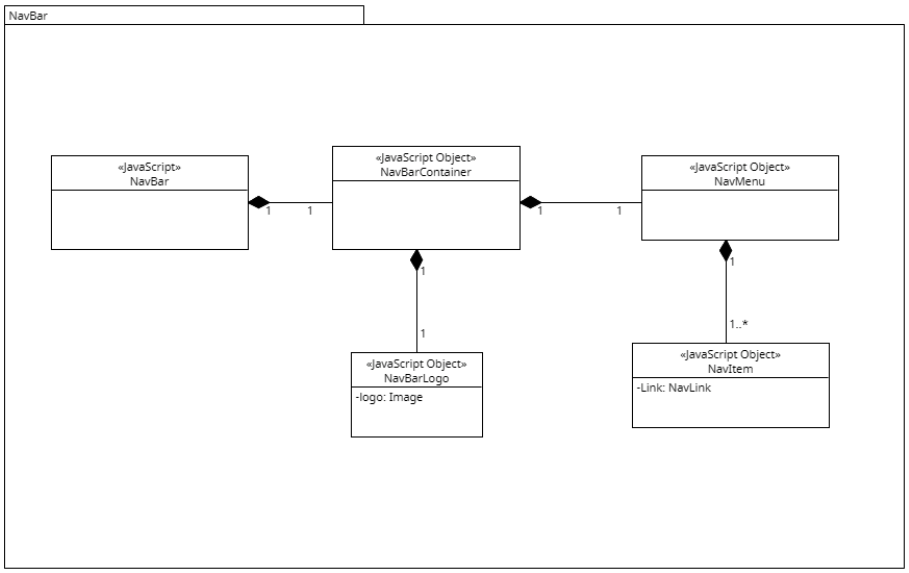


Figure 32: Package NavBar

3.23.1 NavBar

The NavBar is providing the navigation bar.

Declaration

- NavBar

Props

- none

Methods

- none

3.23.2 NavBarContainer

The NavBarContainer is providing the container for the navigation bar.

Declaration

- NavBarContainer

Props

- none

Methods

- none

3.23.3 NavBarLogo

The NavBarLogo is providing the logo for the navigation bar.

Declaration

- NavBarLogo

Props

- private Image logo

Methods

- none

3.23.4 NavMenu

The NavBarMenu is providing the menu for the navigation bar.

Declaration

- NavMenu

Props

- none

Methods

- none

3.23.5 NavItem

The NavItem is providing an item for the navigation bar.

Declaration

- NavItem

Props

- private NavLink link

Methods

- none

3.24 Package AuthorHeader

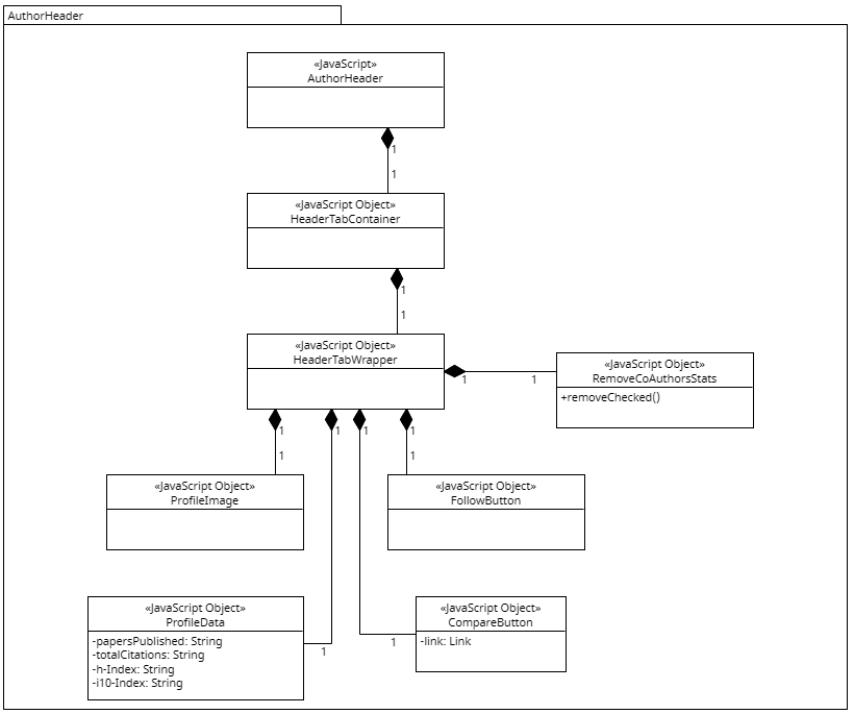


Figure 33: Package AuthorHeader

3.24.1 AuthorHeader

The AuthorHeader is providing the header of an author.

Declaration

- AuthorHeader

Props

- none

Methods

- none

3.24.2 HeaderTabContainer

The AuthorTabContainer is providing the container for a header tab.

Declaration

- HeaderTabContainer

Props

- none

Methods

- none

3.24.3 HeaderTabWrapper

The AuthorTabWrapperr is wrapping the header tab.

Declaration

- HeaderTabWrapper

Props

- none

Methods

- none

3.24.4 ProfileImage

The ProfileImage is providing the profile image of an author.

Declaration

- ProfileImage

Props

- none

Methods

- none

3.24.5 ProfileData

The ProfileData is providing the Information of a profile of an author.

Declaration

- ProfileData

Props

- private String papersPublished
- private String totalCitations
- private String h-index
- private String i10-index

Methods

- none

3.24.6 FollowButton

The FollowButton is providing the follow button on an author's profile.

Declaration

- FollowButton

Props

- none

Methods

- none

3.24.7 CompareButton

The CompareButton is providing the the compare button on an author's profile.

Declaration

- CompareButton

Props

- private Link link

Methods

- none

3.24.8 RemoveCoAuthorsStats

The RemoveCoAuthorsStats is providing the option to remove the stats of co-authors.

Declaration

- CompareButton

Props

- private Link link

Methods

- public void removeChecked()

3.25 Package SearchBar

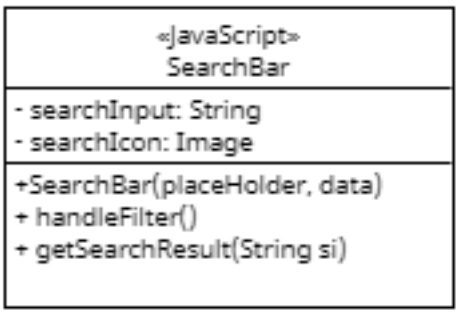


Figure 34: Package SearchBar

3.25.1 Searchbar

The SearchBar is providing the search bar.

Declaration

- Searchbar

Props

- private String searchInput
- private String searchIcon

Methods

- public void searchBar(placeHolder, data
- public void handleFilter()
- public void getSearchResult(String si

3.26 Package PublicationCard

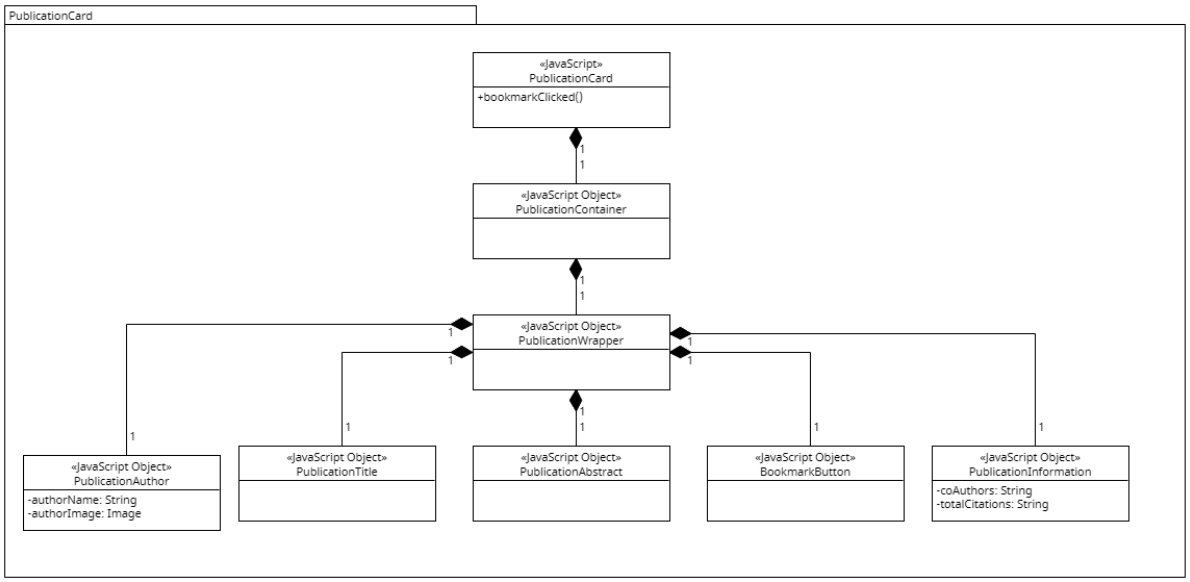


Figure 35: Package PublicationCard

3.26.1 PublicationCard

The PublicationCard is providing the card of publications.

Declaration

- PublicationCard

Props

- none

Methods

- public boookmarkClicked()

3.26.2 PublicationContainer

The PublicationContainer is providing the container for the publications.

Declaration

- PublicationContainer

Props

- none

Methods

- none

3.26.3 PublicationWrapper

The PublicationWrapper is wrapping the publications.

Declaration

- PublicationWrapper

Props

- none

Methods

- none

3.26.4 PublicationAuthor

The PublicationAuthor is providing the author of the publication.

Declaration

- PublicationAuthor

Props

- private String authorName
- private Image authorImage

Methods

- none

3.26.5 PublicationTitle

The PublicationTitle is providing the title of the publication.

Declaration

- PublicationTitle

Props

- none

Methods

- none

3.26.6 PublicationAbstract

The PublicationAbstract is providing the abstract of the publication.

Declaration

- PublicationAbstract

Props

- none

Methods

- none

3.26.7 BookmarkButton

The BookmarkButton is providing the button to bookmark the publication.

Declaration

- BookmarkButton

Props

- none

Methods

- none

3.26.8 PublicationInformation

The PublicationInformation is providing the information of the publication.

Declaration

- PublicationInformation

Props

- private String coAuthors
- private String totalCitations

Methods

- none

4 Database design

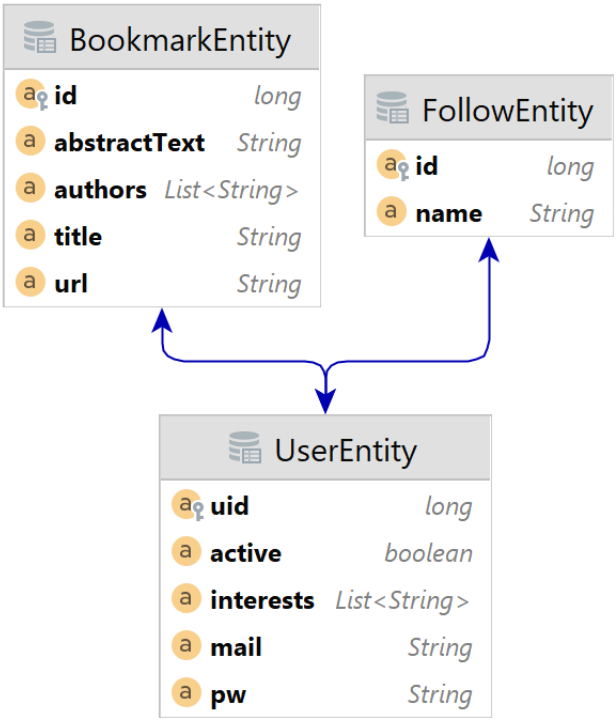


Figure 36: Database design diagram containing tables

Here every UserEntity has a list of FollowEntities and a list of BookmarkEntities, annotated with @ManyToMany and a @JoinTable joining the two respective tables, respectively called follows and bookmarks. The FollowEntity and BookmarkEntity both have a list od UserEntities annotated with the respective @ManyToMany to confirm the mapping. JPA takes care of connecting these tables with a mapping table, of which the names can be defined in the @ManyToMany annotation in the UserEntity class.

5 Sequence and Activity diagrams

5.1 Activity diagrams

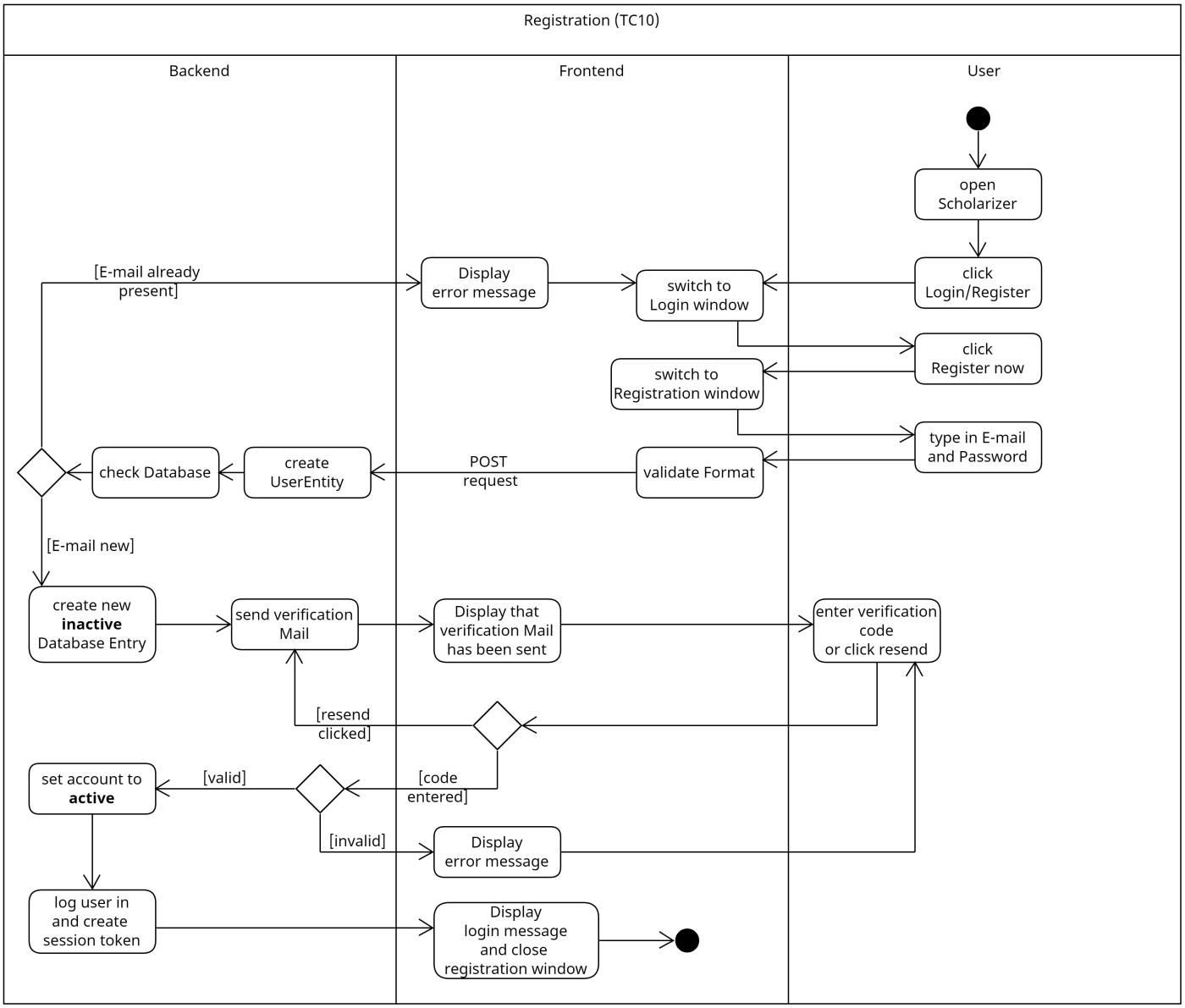


Figure 37: Registration

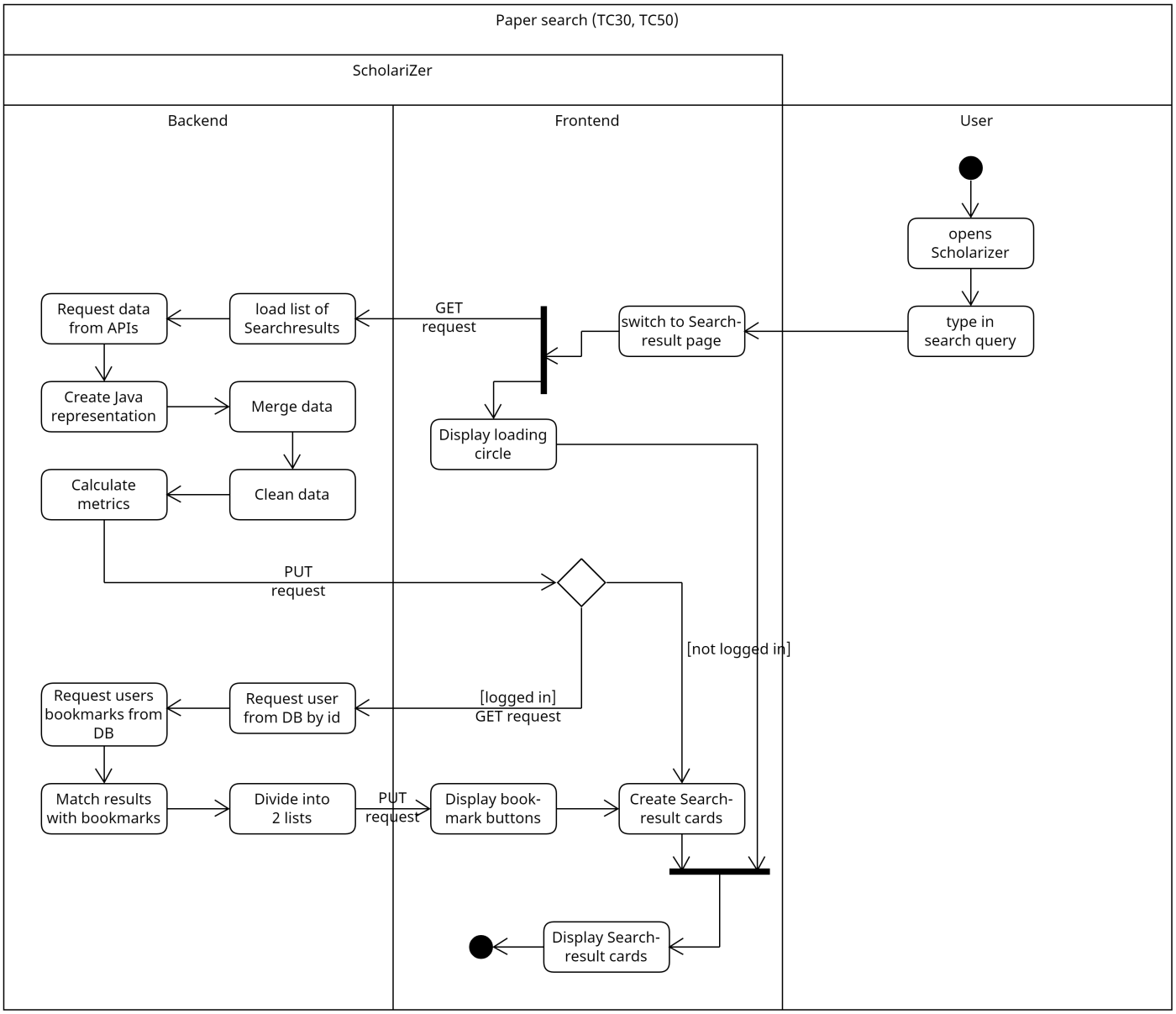


Figure 38: Paper search

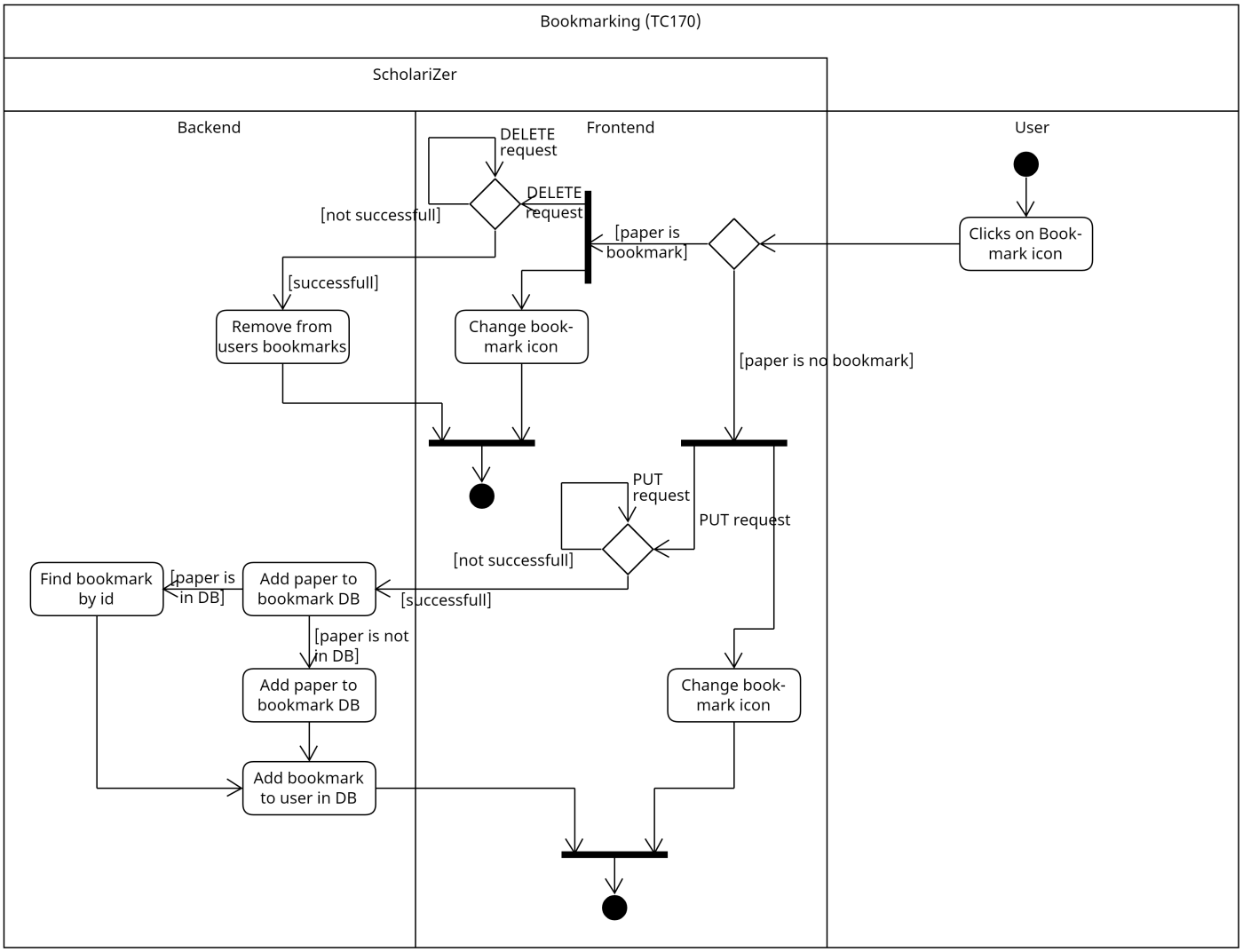


Figure 39: Bookmarking

5.2 Sequence diagrams

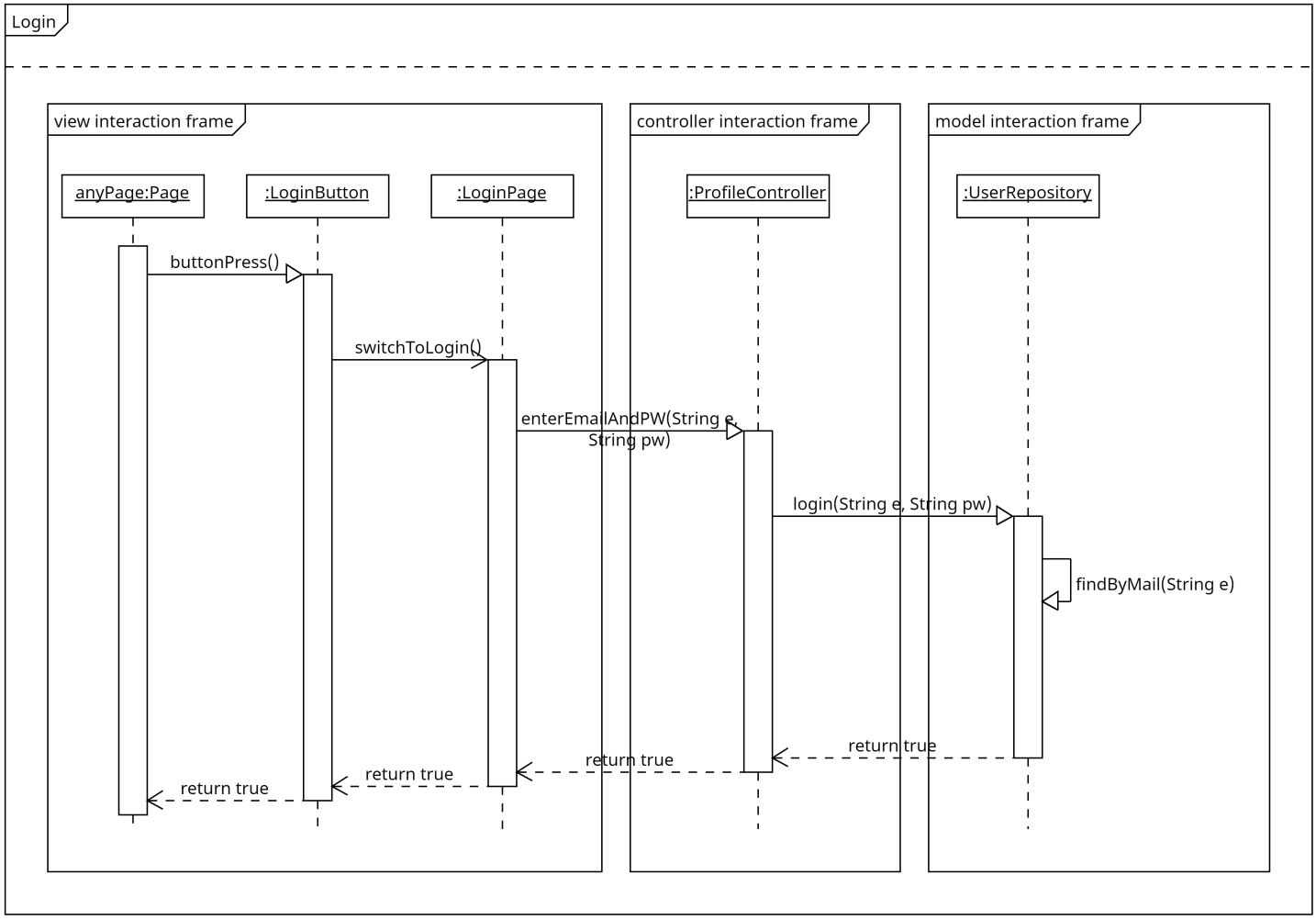


Figure 40: Login

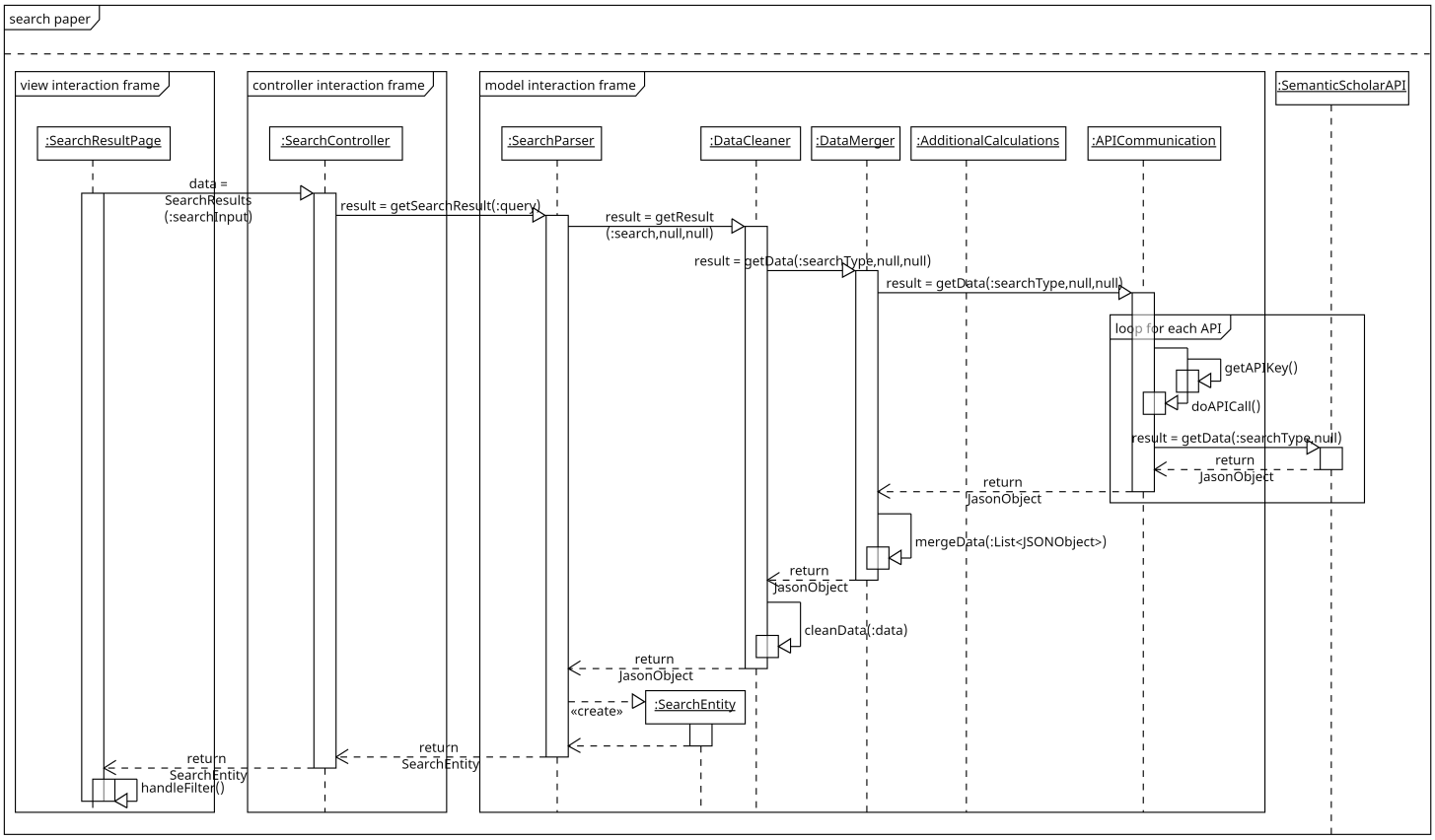


Figure 41: Paper search

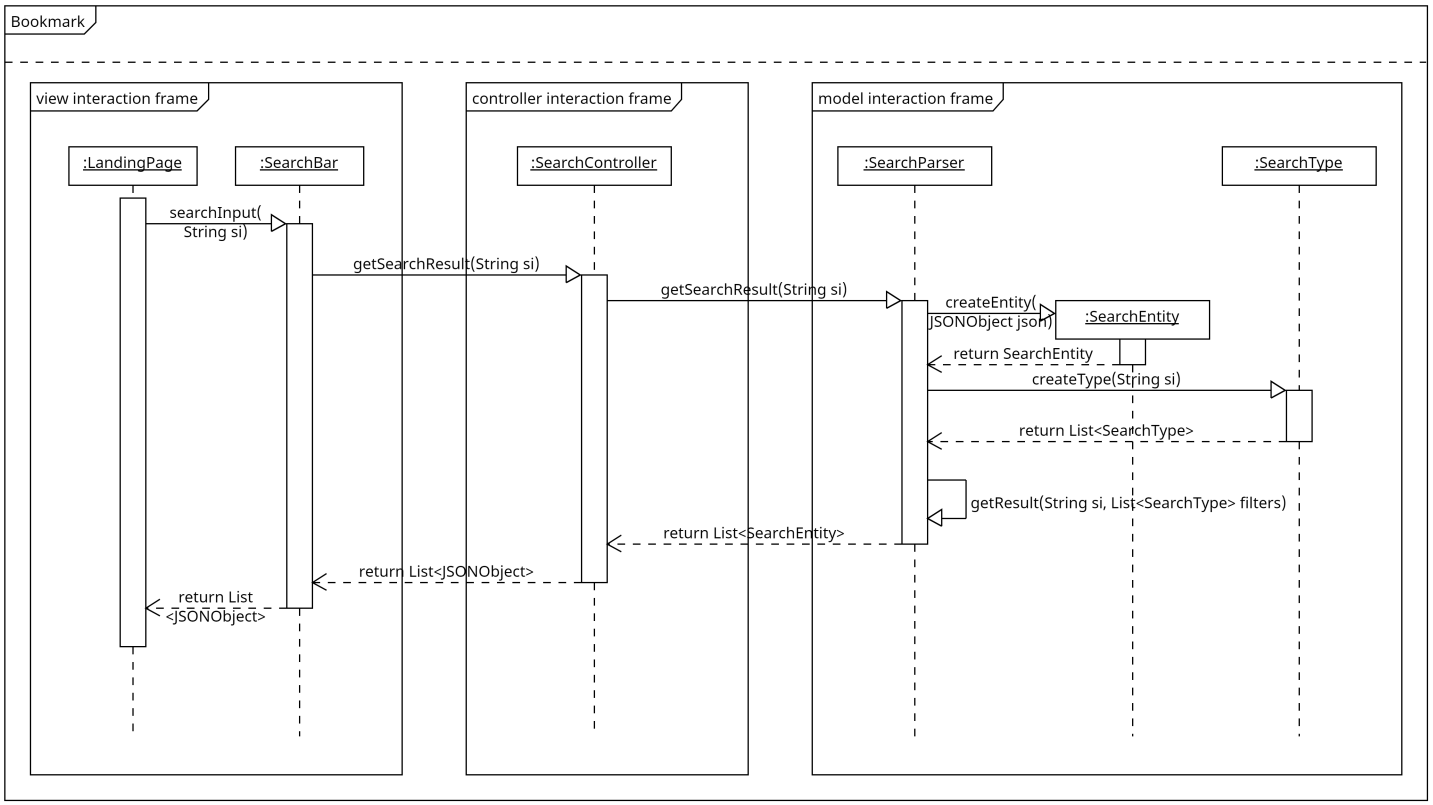


Figure 42: Bookmarking

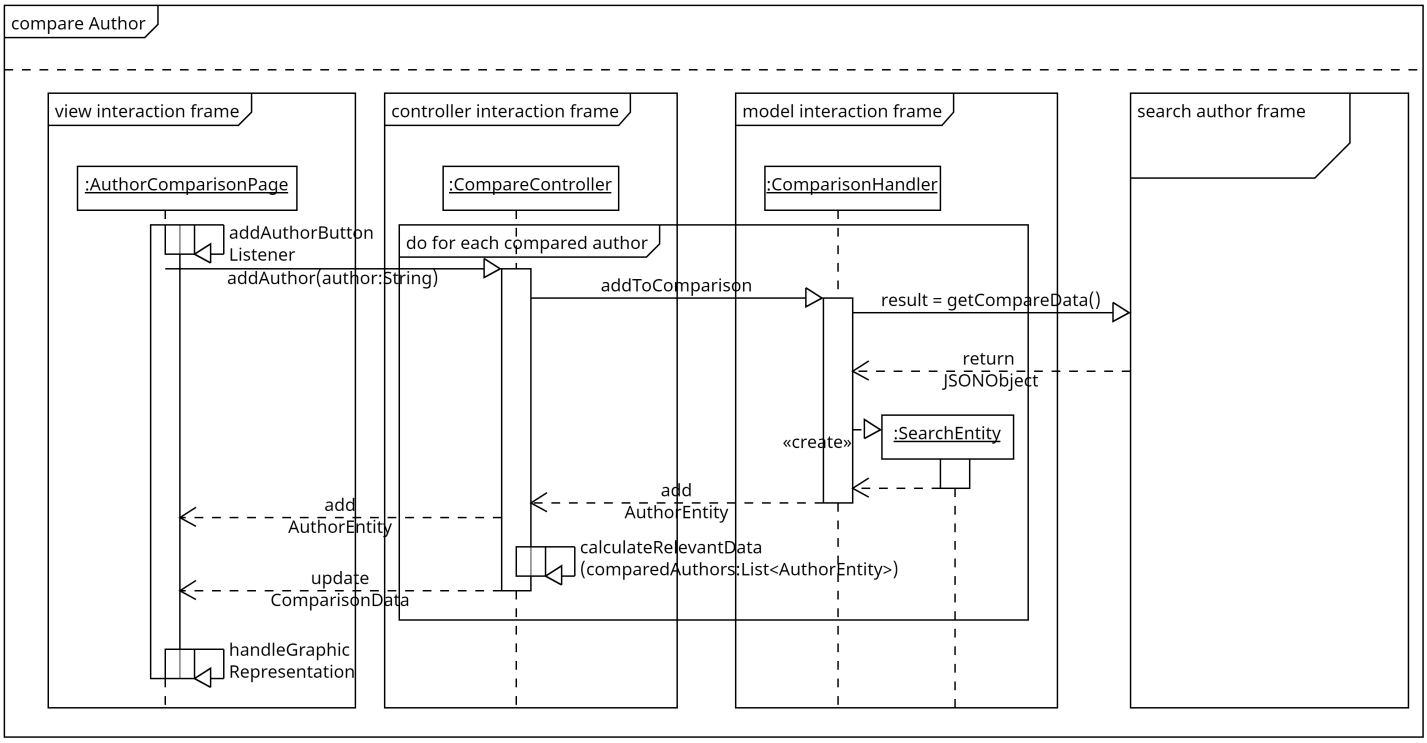


Figure 43: Compare authors

(For closer look at *search author frame*, see figure 41.)

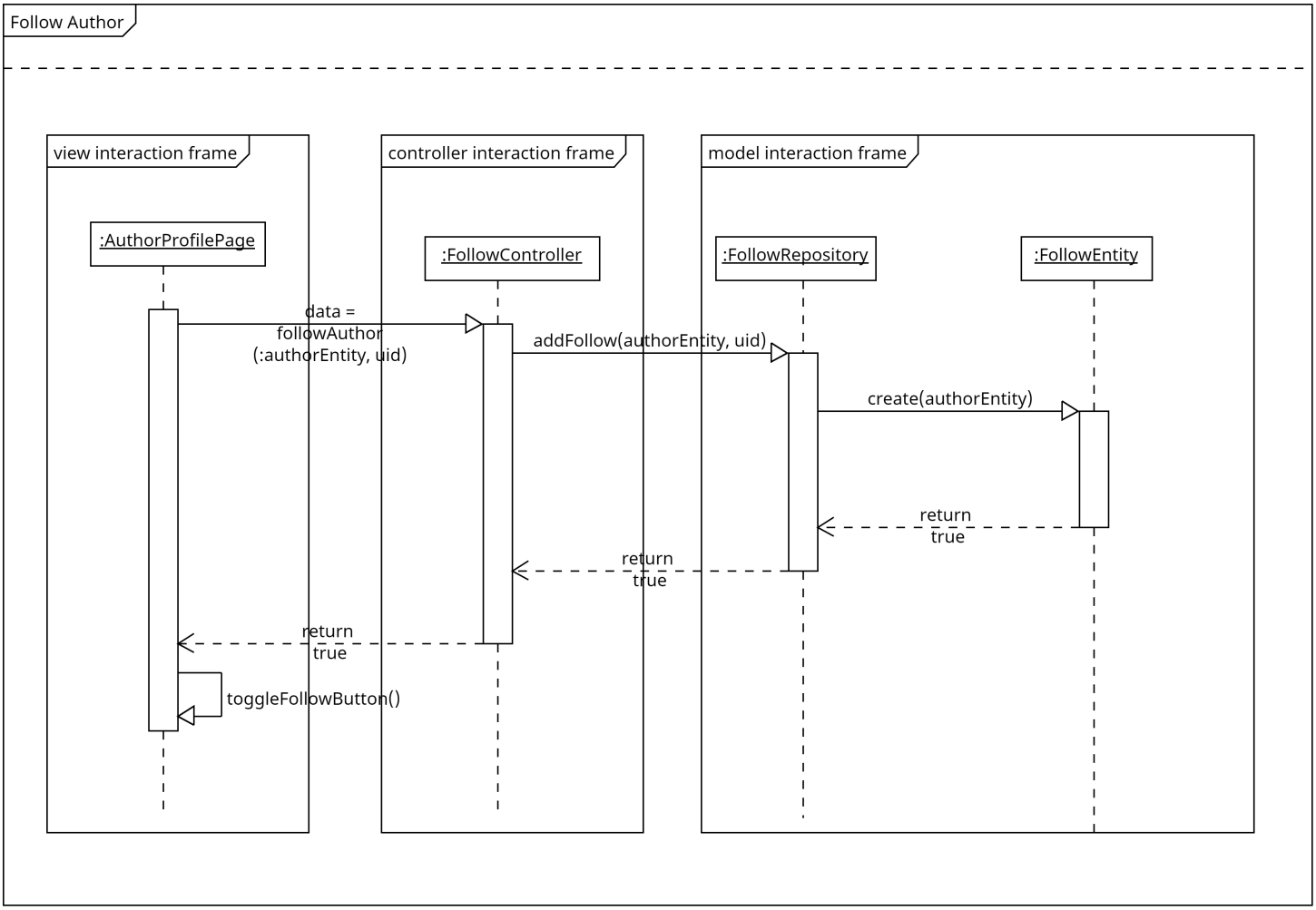


Figure 44: Follow

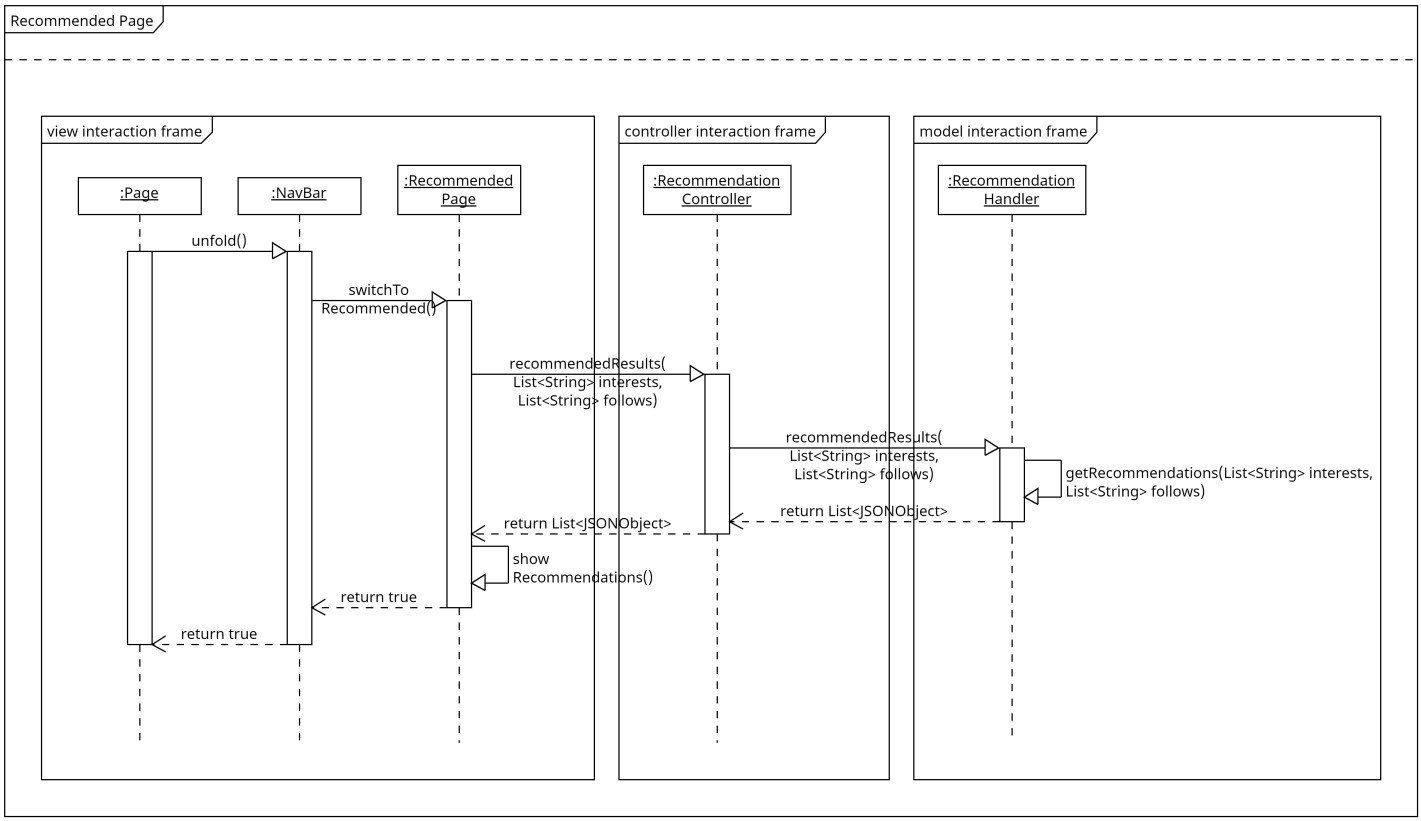
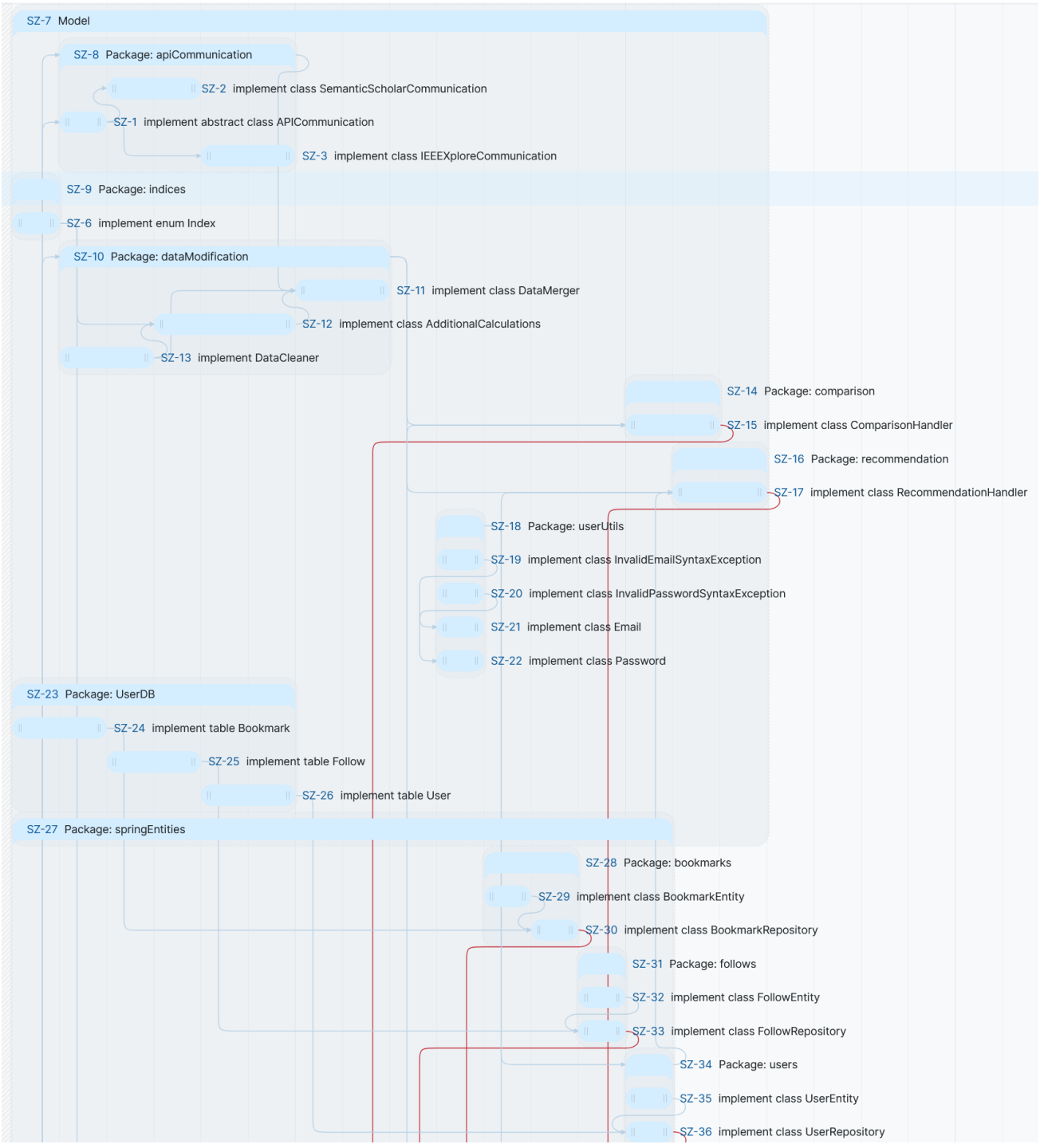
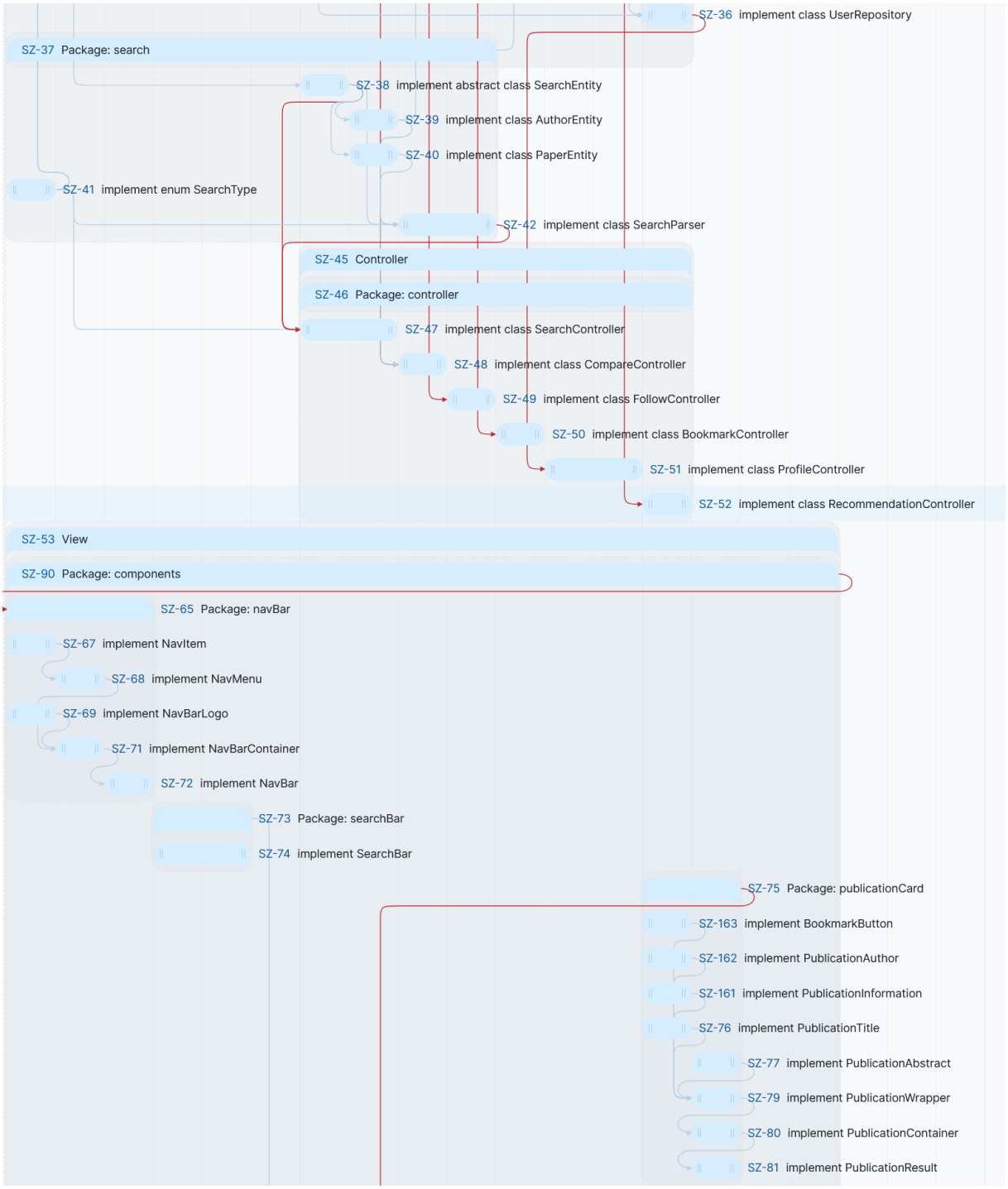
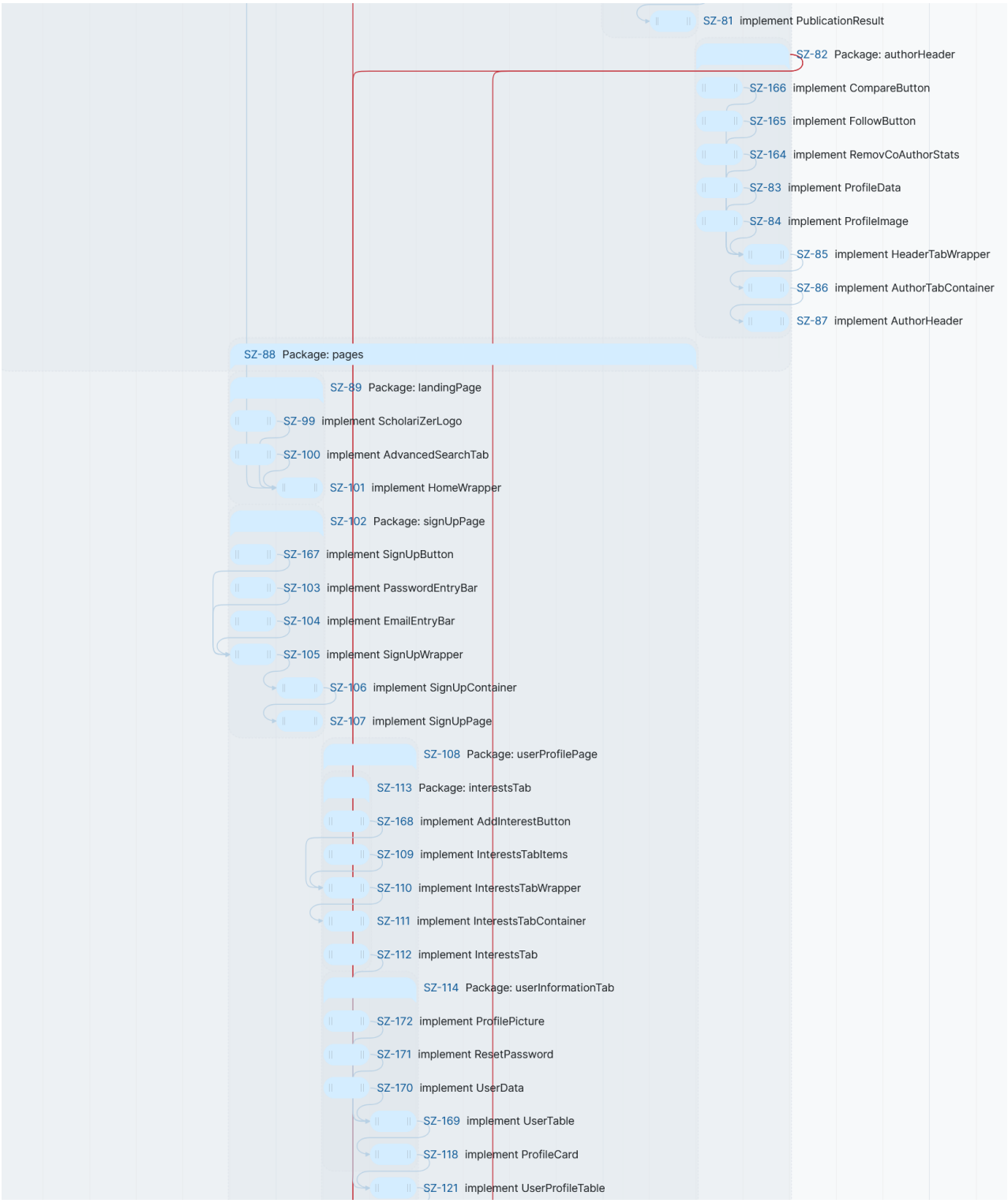


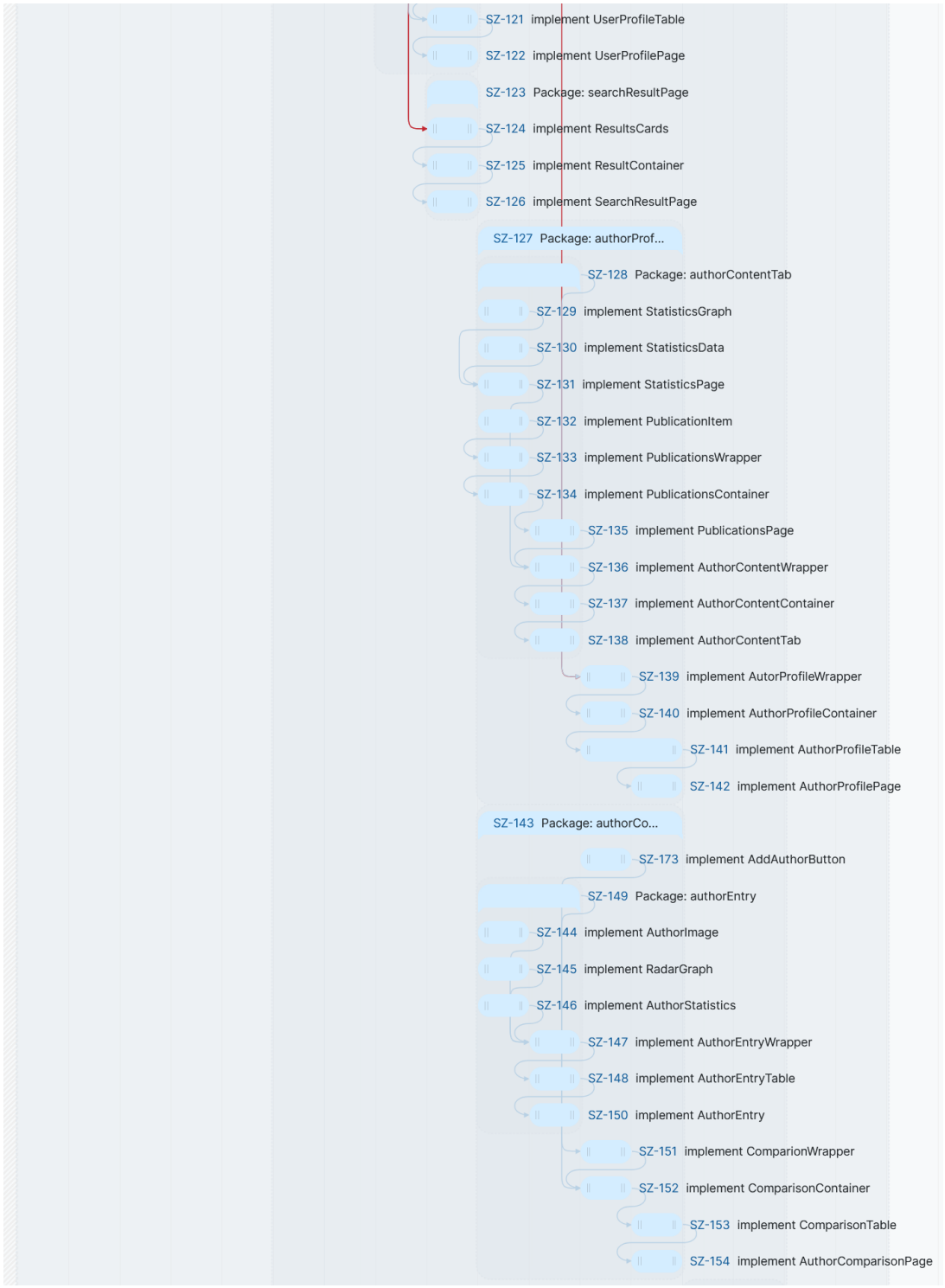
Figure 45: Recommendation

6 Gantt chart




SZ-36 implement class UserRepository
SZ-37 Package: search
SZ-38 implement abstract class SearchEntity
SZ-39 implement class AuthorEntity
SZ-40 implement class PaperEntity
SZ-41 implement enum SearchType
SZ-42 implement class SearchParser
SZ-45 Controller
SZ-46 Package: controller
SZ-47 implement class SearchController
SZ-48 implement class CompareController
SZ-49 implement class FollowController
SZ-50 implement class BookmarkController
SZ-51 implement class ProfileController
SZ-52 implement class RecommendationController
SZ-53 View
SZ-90 Package: components
SZ-65 Package: navBar
SZ-67 implement NavItem
SZ-68 implement NavMenu
SZ-69 implement NavBarLogo
SZ-71 implement NavBarContainer
SZ-72 implement NavBar
SZ-73 Package: searchBar
SZ-74 implement SearchBar
SZ-75 Package: publicationCard
SZ-163 implement BookmarkButton
SZ-162 implement PublicationAuthor
SZ-161 implement PublicationInformation
SZ-76 implement PublicationTitle
SZ-77 implement PublicationAbstract
SZ-79 implement PublicationWrapper
SZ-80 implement PublicationContainer
SZ-81 implement PublicationResult


SZ-81 implement PublicationResult
SZ-82 Package: authorHeader
SZ-166 implement CompareButton
SZ-165 implement FollowButton
SZ-164 implement RemovCoAuthorStats
SZ-83 implement ProfileData
SZ-84 implement ProfileImage
SZ-85 implement HeaderTabWrapper
SZ-86 implement AuthorTabContainer
SZ-87 implement AuthorHeader
SZ-88 Package: pages
SZ-89 Package: landingPage
SZ-99 implement ScholariZerLogo
SZ-100 implement AdvancedSearchTab
SZ-101 implement HomeWrapper
SZ-102 Package: signUpPage
SZ-167 implement SignUpButton
SZ-103 implement PasswordEntryBar
SZ-104 implement EmailEntryBar
SZ-105 implement SignUpWrapper
SZ-106 implement SignUpContainer
SZ-107 implement SignUpPage
SZ-108 Package: userProfilePage
SZ-113 Package: interestsTab
SZ-168 implement AddInterestButton
SZ-109 implement InterestsTabItems
SZ-110 implement InterestsTabWrapper
SZ-111 implement InterestsTabContainer
SZ-112 implement InterestsTab
SZ-114 Package: userInformationTab
SZ-172 implement ProfilePicture
SZ-171 implement ResetPassword
SZ-170 implement UserData
SZ-169 implement UserTable
SZ-118 implement ProfileCard
SZ-121 implement UserProfileTable


SZ-121 implement UserProfileTable
SZ-122 implement UserProfilePage
SZ-123 Package: searchResultPage
SZ-124 implement ResultsCards
SZ-125 implement ResultContainer
SZ-126 implement SearchResultPage
SZ-127 Package: authorProf...
SZ-128 Package: authorContentTab
SZ-129 implement StatisticsGraph
SZ-130 implement StatisticsData
SZ-131 implement StatisticsPage
SZ-132 implement PublicationItem
SZ-133 implement PublicationsWrapper
SZ-134 implement PublicationsContainer
SZ-135 implement PublicationsPage
SZ-136 implement AuthorContentWrapper
SZ-137 implement AuthorContentContainer
SZ-138 implement AuthorContentTab
SZ-139 implement AutorProfileWrapper
SZ-140 implement AuthorProfileContainer
SZ-141 implement AuthorProfileTable
SZ-142 implement AuthorProfilePage
SZ-143 Package: authorCo...
SZ-173 implement AddAuthorButton
SZ-149 Package: authorEntry
SZ-144 implement AuthorImage
SZ-145 implement RadarGraph
SZ-146 implement AuthorStatistics
SZ-147 implement AuthorEntryWrapper
SZ-148 implement AuthorEntryTable
SZ-150 implement AuthorEntry
SZ-151 implement ComparionWrapper
SZ-152 implement ComparisonContainer
SZ-153 implement ComparisonTable
SZ-154 implement AuthorComparisonPage

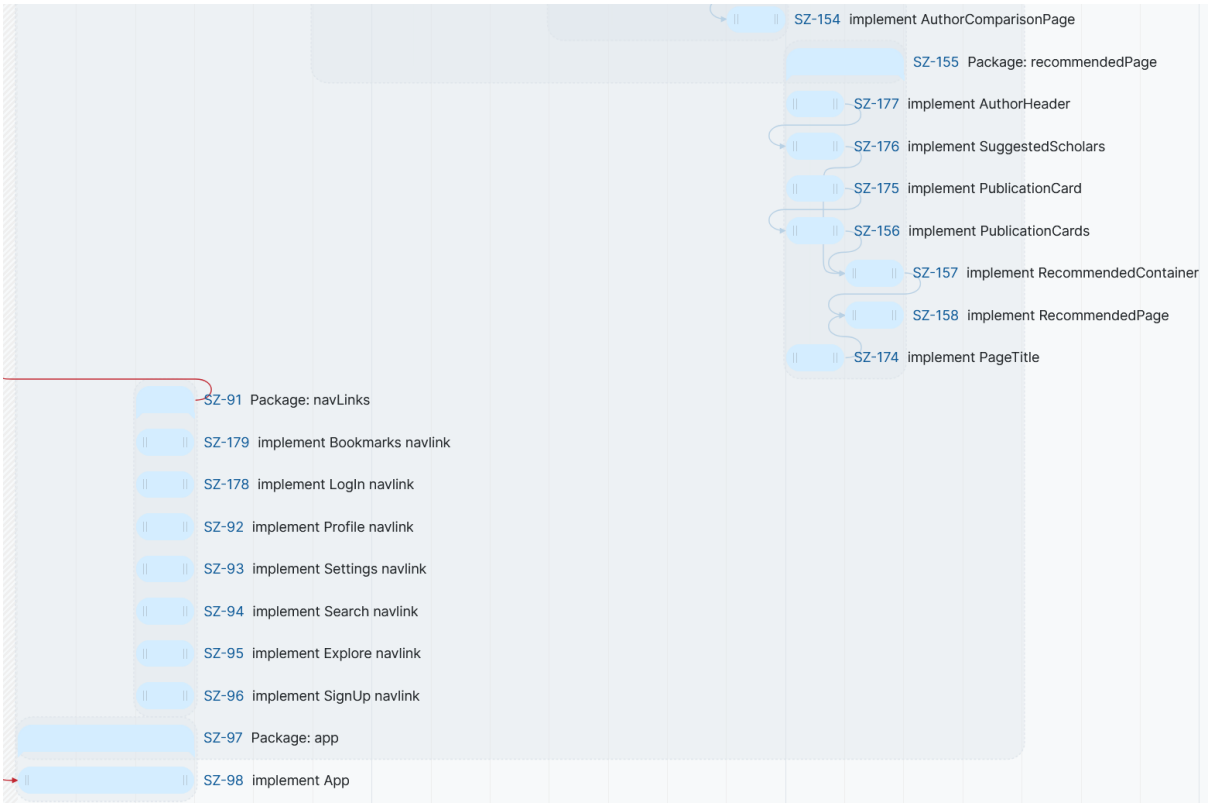


Figure 46: Gantt chart: Implementation plan

The Gantt chart shows tasks displayed against time, visualizing the implementation plan for the ScholariZer webapp. Each bar represents a task to implement a package, class, an entity or a table. The relationships between parent tasks and subtasks are represented by two vertical lines. The line on the left represents the start date of the earliest subtask. The line on the right represents the end date of the latest subtask. The relationship between dependent tasks is represented by an arrow. The arrow is drawn from the end date of the predecessor task to the start date of the successor task. Red arrows indicate unresolved predecessors. These occur as a result of a parallelized implementation process and will be resolved using test doubles.

6.1 Implementation Dashboard

6.1.1 Assigned to Dominik Jentsch

Figure 47: Implementation assigned to Dominik Jentsch

6.1.2 Assigned to Tim Wolk

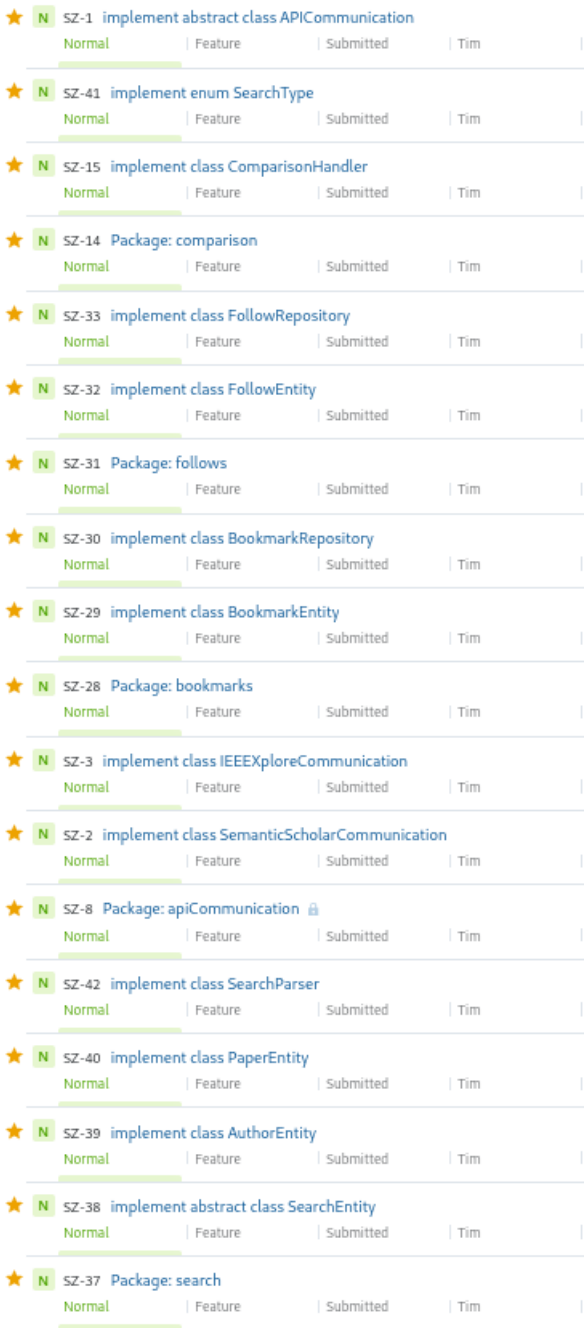


Figure 48: Implementation assigned to Tim Wolk

6.1.3 Assigned to Alexander Möhring

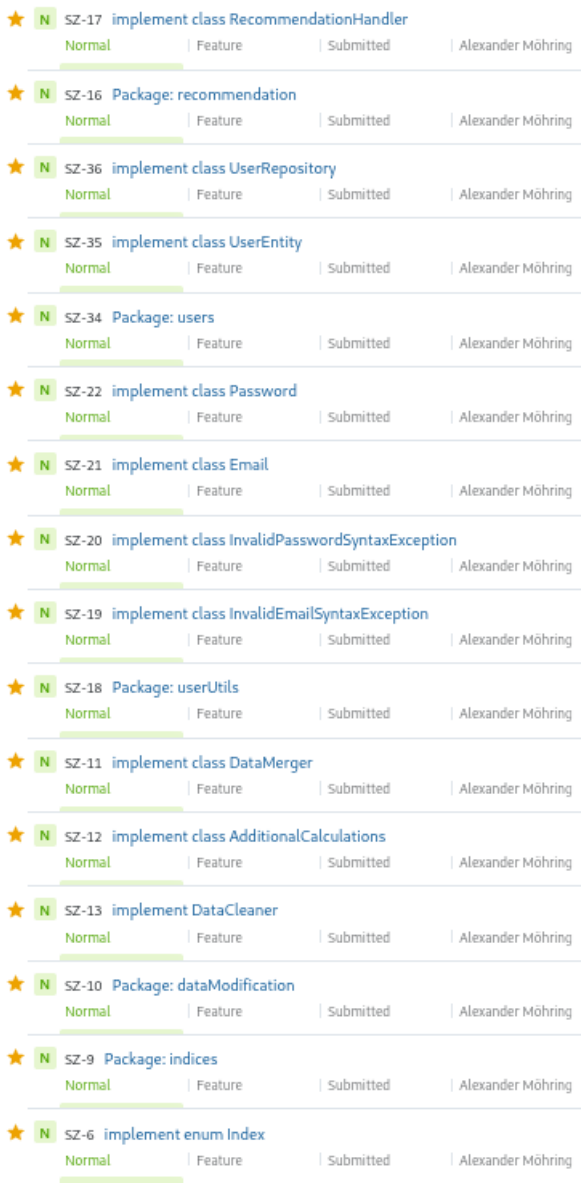


Figure 49: Implementation assigned to Alexander Möhring

6.1.4 Assigned to Pablo Schmeiser

Figure 50: Implementation assigned to Schmeiser

6.1.5 Assigned to Adham Gouda

Figure 51: Implementation assigned to Adham Gouda

7 Tools and Technologies

- Database: PostgreSQL
- APIs: Semantic Scholar and maybe IEEE-Xplore
- Server: A KIT system
- Server Operating System: A Linux distribution with this projects Docker container running on top of it.
- Container management: Docker, Docker Desktop and maybe kubernetes
- IDEs: IntelliJ, Webstorm and Visual Studio Code
- Frontend Languages: TypeScript 4.9 with Babel as a transcompiler
- Frontend Frameworks: Next.js 13.0.5 and React.js 18.2.0
- Frontend Libraries: Material UI
- Backend Languages: Java (OpenJDK 19) built using Maven 3.8
- Backend Frameworks: Spring using Spring Initializr
- Spring Dependencies: Spring Web, Rest Repositories, Spring Security, Spring Data JPA, PostgreSQL Driver, Java Mail Sender, Spring REST Docs and maybe Spring Boot Actuator
- UML diagrams: Lucid Chart, UMLet 15.0 via it's Visual Studio Code Plug-In
- Version Control: GitHub via Git
- Package Managers: npm and Maven
- Organization and productivity tools: Notion, Google Calendar and YouTrack by JetBrains
- Code Quality inspection: Sonarqube, SonarLint, ESLint and Checkstyle
- Testing: JUnit, Jest, Postman and maybe BrowserStack's Cypress
- Webbrowsers to view GUI: Firefox and Google Chrome
- Writing: Overleaf to write LaTeX
- Documentation: Javadoc, README in Markdown