RehabStar Overview

May 1, 2018

1 Back-End API

RehabStar has four domain objects; a User, Story, FollowingPair, and a ConnectingPair. The FollowingPair connects two users where a ConnectingPair connects a user to a story. Each User object has a user name, password, and email address. To distinguish between Users in the database an ID is assigned to each user which is an integer value. There are also fields for users to keep track of their days clean and keep a goal(some amount of days clean). Stories also have integer ID's used to retrieve them from the database. Stories can also be assigned three keywords that are compared when searching stories, and can also be found via the date created.

Each object has a data access object associated with it which contains methods to interact with the database. There are interfaces and implementations for each domain object. The next layer up is the services layer which connects the data access layer to the controllers. The controllers provide the paths to be called by the application's user interface.

1.1 SQL Tables

There are four tables used in the database. These tables are; a User table, a Story table, a Connections Table, and a Following table which store their respective objects.

1.2 Domain Objects

1.2.1 User

- Constructors

  public User()

  public User(String userName, String email, String password)

  public User(Integer id, String userName, String email, String password)

  public User(int id, String userName, String email, String password, int daysClean, int goalDaysClean)

- Fields

  private int id: integer, unique to the user to find them in the database

  private String username: String, username set by user

  private String email: String, user email address

  private String password: String, user password

  private int daysClean: integer, number of consecutive days clean

  private int goalDaysClean: integer, set by the user as a goal amount of days

  private String currentSearch: String, search fields that can be set so that the user can find stories

1.2.2 Story

The Story object contains 13 fields. 3 are keywords to be used in finding the story via corresponding topics. There is a title field and also an ID field, used to distinguish a Story from others. A Story also holds the time and date which it was created and the ID of the creator. Finally, there is a field to store a .txt file of the Story. There are also getters and setters for each field.

- Constructors

  public Story()

  public Story(int userId, String fileName, String title, Timestamp dateCreated, int likes)

  public Story(int userId, String fileName, String title, byte[] text, Timestamp dateCreated)

  public Story(int userId, String fileName, String title, byte[] text, Timestamp dateCreated, String keyword1, String keyword2, String keyword3, int likes)

public Story(int id, int userId, String fileName, String title, byte[] text, Timestamp dateCreated, String keyword1, String keyword2, String keyword3, int likes)

public Story(int id, int userId, String fileName, String title, Timestamp dateCreated, int likes)

public Story(int id, int userId, String fileName, String title, byte[] text, Timestamp dateCreated)

- Fields

  private int id; a unique ID to the Story

  private int userId; the users ID who created the Story

  private String fileName; a filename extension for the Story

  private String title; title of the Story

  private byte[] text;

  private String plainText;

  private String userName; username that created the Story

  private Timestamp dateCreated;

  private String time;

  private String keyword1; Keywords to allow the Story to be searched for

  private String keyword2;

  private String keyword3;

  private int likes; amount of "likes" or "follows"

1.2.3 FollowingPair

The FollowingPair object holds two values; the integer ID of a User and the integer ID of the User whom the first is following.

- Constructors

  public FollowingPair()

  public FollowingPair(int userId, int followingId)

- Fields

  private int userId;

  private int followingId;

1.2.4 ConnectionPair

The ConnectionPair is similar to the following pair but instead holds the ID of a User and the ID of a Story.

- Constructors

  public ConnectionPair()

  public ConnectionPair(int userId, int storyId)

- Fields

  private int userId;

  private int storyId;

1.3 Data Access Layer

1.3.1 UserDAO

The UserDAO interacts with the SQL table and contains methods for accessing and manipulating the User object. One can pull a User by name from the table. There are also methods for finding the days clean and goal days clean for a particular userID as well as updating the password or email.

- Methods

  public User findUserByUserName(String userName)

  public void addUser(User u)

  public void updateUserName(int id, String userName)

  public void updateEmail(int id, String email)

  public void updatePassword(int id, String password)

  public void deleteUser (int id)

  public void incrementDaysClean (int userId)

  public int findDaysCleanById (int userId)

  public int findGoalDaysCleanById (int userId)

  public void setGoalDaysCleanById(int userId, int set)

1.3.2 StoryDAO

Using the StoryDAO, one can access or change various fields of a Story object as well as find a User from the Story. One can also return all Stories, titles, and also return a number of stories within a given amount of days/time.

- Methods

  int findUserIdByStoryId(int storyId)

  Story findStoryById(int storyid)

  List<Story>findStoriesByUserId(int userId)

  byte[] findTextByStoryId(int id)

  String findFileNameByStoryId(int id)

  List<Story>findAllStories()

  void updateStoryText(int storyId, byte [] text)

  void addStory(Story s)void addStory(Story s)

  void deleteStory(Story s)

  List<Story>findStoriesByTitle(String title)

  List<String>findAllTitles(int userId)

  Timestamp findDateCreatedById(int storyId)

  List<Story>findStoriesWithinDays(int daysSince)

  List<Story>findStoriesWithinHours(int hoursSince)

  List<Story>findStoriesByAKeyword(String keyword, int userId)

  void updateKeywordsById(int id, String keyword1, String keyword2, String keyword3)

  List<Story>findOneUserStoriesWithinDays(int userId, int daysSince)

  List<Story>findAllStoriesNotUsers(int userId)

  List<Story>findAllStoriesNotUsersWithinDays(int userId, int daysSince)

  void likeStory(int storyId)

1.3.3 FollowingPairDAO

- Methods

  List<FollowingPair>findFollowerIds(int userId)

  void addFollowingPair(int userId, int toFollowId)

1.3.4 ConnectionPairDAO

- Methods

  List<ConnectionPair>findConnectionPairsForUser(int userId)

  void addConnectionPair(int userId, int storyId)

1.4 Services Layer

Each DAO contains its own service class which calls on the methods and allows for a much cleaner and easier to read interface to be used by the controllers. There is also a service for a forgotten password which allows for its retrieval.

1.4.1 UserService

- Methods

  List<User>setGoalDaysCleangetAllUsers()

  void createUser(String userName, String email, String password)

  void updateUserName(int id, String userName)

  void updateEmail(int id, String email)

  void updatePassword(int id, String password)

  void deleteUser(int id)

  User findUserById(int id)

  User findUserByUserName(String userName)

  boolean authenticate(String userName, String password)

  void incrementDaysClean(int id)

  void setGoalDaysClean(int id, int goal)

1.4.2 StoryService

- Methods

  int findUserIdByStoryId(int storyId)

  Story findStoryById(int storyid)

  List<Story>findStoriesByUserId(int userId)

  byte[] findTextByStoryId(int id)

  List<Story>findAllStories()

  void updateStoryText(int storyId, byte[] text)

  void addStory(Story s)

  void deleteStory(Story s)

  List<Story>findStoriesByTitle(String title)

  List<Story>findStoriesByTitleSubstring(String substring, int userId)

  Timestamp findDateCreatedById(int storyId)

  List<Story>findStoriesWithinDays(int daysSince)

  List<Story>findStoriesWithinHours(int hoursSince)

  List<Story>sortStoriesForMostRecent(List¡Story¿ stories)

  List<Story>findStoriesByAKeyword(String keyword, int userId)

  void updateKeywordsById(int id, String keyword1, String keyword2, String keyword3)

  List<Story>findOneUserStoriesWithinDays(int userId, int daysSince)

  List<Story>findAllStoriesNotUsers(int userId)

  List<Story>findAllStoriesNotUsersWithinDays(int userId, int daysSince)

  String convertToPlainText(byte[] bytes)throws java.io.UnsupportedEncodingException

  byte[] convertTextToBytes(String text)

  void likeStory(int storyId)

  List<Story>sortStoriesForMostConnections()

1.4.3 FollowingPairService

- Methods

  List<FollowingPair>findFollowerIds(int userId)

  void addFollowingPair(int userId, int toFollowId)

1.4.4 ConnectionPairService

- Methods

  List<ConnectionPair>findConnectionPairsForUser(int userId)

  void addConnectionPair(int userId, int storyId)

1.4.5 ForgotPassword

- Methods

  void Forgot(String email, String userName)throws IOException

1.5 Controller Layer

1.5.1 User Controller

Using the User controller, the UI can call on various mappings to update the fields of a User object and return who he/she is following.

- Constructor

  public UserController(UserService userService, ForgotPassword forgotPassword)

- Methods

  @RequestMapping(value = "/findAllUsers")
  public @ResponseBody List<User>findAllUser()

  @RequestMapping(value = "/findUserById/id", method = RequestMethod.GET)
  public @ResponseBody User findUserById(@PathVariable("id") int id)

  @RequestMapping(value = "/updateUserName/id/username", method = RequestMethod.GET)
  public @ResponseBody User updateUserNameById(@PathVariable("id") int id, @PathVariable("username") String username)

RequestMapping(value = "/updateUserEmail/id/email", method = RequestMethod.GET)

public @ResponseBody User updateUserEmailById(@PathVariable("id") int id, @PathVariable("email") String email)

@RequestMapping(value = "/updateUserPassword/id/password", method = RequestMethod.GET)

public @ResponseBody User updateUserPasswordById(@PathVariable("id") int id, @PathVariable("password") String password)

@RequestMapping(value = "/findUserByUserName/username", method = RequestMethod.GET)

public @ResponseBody User findUserByUserName(@PathVariable("username") String username)

@RequestMapping(value = "/authenticate/userName/password", method = RequestMethod.GET)

public @ResponseBody boolean authenticate(@PathVariable("userName") String userName, @PathVariable("password")

@RequestMapping(value = "/incrementDaysCleanById/id", method = RequestMethod.GET)

public @ResponseBody void incrementDaysClean(@PathVariable int id)

@RequestMapping(value = "/setGoalDaysCleanById/id/goal", method = RequestMethod.GET)

public @ResponseBody void setGoalDaysClean(@PathVariable int id, @PathVariable int goal)

@RequestMapping(value = "/forgotPassword/email/userName", method = RequestMethod.GET)

public @ResponseBody void Forgot(@PathVariable String email, @PathVariable String userName)

@RequestMapping(value = "/createUser/username/email/password", method = RequestMethod.POST)

public void addUser(@PathVariable String email, @PathVariable String username, @PathVariable String password) userService.createUser(username, email, password)

1.5.2 Story Controller

The Story controller provides mappings to update and return Story objects. Using this controller one can also download or upload a file of Stories to the database.

- Constructor

  public StoryController(StoryService storyService, UserService userService)

- Methods

  @RequestMapping(value = "/findAllStories")

  public @ResponseBody List<Story>findAllStories()

@RequestMapping(value = "/updateStoryText/id/text", method = RequestMethod.GET)

public @ResponseBody Story updateStoryNameById(@PathVariable("id") int id, @PathVariable("text") String text)

@RequestMapping(value = "/findStoryById/id", method = RequestMethod.GET)

public @ResponseBody Story findStoryById(@PathVariable("id") int id)

@RequestMapping(value = "/findStoriesByUserId/userId", method = RequestMethod.GET)

public @ResponseBody List<Story>findStoriesByUserId(@PathVariable("userId") int userId)

RequestMapping(value = "/findStoriesByTitleSubstring/sub/userId", method = RequestMethod.GET)

public @ResponseBody List<Story>findStoriesByTitleSubstring(@PathVariable("sub") String sub, @PathVariable("userId") int userId)

@RequestMapping(value = "/findStoriesWithinDays/daysSince", method = RequestMethod.GET)

public @ResponseBody List<Story>findStoriesWithinDays(@PathVariable("daysSince") int daysSince)

@RequestMapping(value = "/findStoriesWithinHours/hoursSince", method = RequestMethod.GET)

public @ResponseBody List<Story>findStoriesWithinHours(@PathVariable("hoursSince") int hoursSince)

@RequestMapping(value = "sortUserStoriesForMostRecent/id")

public @ResponseBody List<Story>sortStoriesForMostRecent(@PathVariable ("id") int storyId)

@RequestMapping(value = "findStoriesByAKeyword/keyword/userId")

public @ResponseBody List<Story>findStoriesByAKeyword(@PathVariable ("keyword") String keyword, @PathVariable ("userId") int userId)

1.5.3 FollowingPair Controller

The FollowingPair Controller provides mappings to create or return new pairs of users to users.

- Constructors

  public FollowingPairController(FollowingPairService followingPairService, UserService userService)

- Methods

  @RequestMapping(value = "/findUserFollowersId/id", method = RequestMethod.GET)

  public @ResponseBody List<FollowingPair>findUserFollowerId(@PathVariable("id") int id)

@RequestMapping(value = "/addFollower/id", method = RequestMethod.POST)

public void addFollower(@ModelAttribute(value = "user") User user, @PathVariable("id") int id)

1.5.4 ConnectionPair Controller

The ConnectionPair Controller provides mappings to create or return new pairs of users to stories.

- Constructors

  public ConnectionPairController(ConnectionPairService connectionPairService, StoryService storyService)

- Methods

  @RequestMapping(value = "/findStoriesThatUserConnectedTo", method = RequestMethod.GET)

  public @ResponseBody List<Story>findFeedForUser(@ModelAttribute("user") User user)

  @RequestMapping(value = "/connectTo/storyId", method = RequestMethod.GET)

  public void connectToAStory(@ModelAttribute("user") User user, @PathVariable("storyId") int storyId)

1.5.5 Feed Controller

The Feed controller displays a list of stories given a time window. Currently it displays ten, if the User does not have 10 updates of other Users he/she is following it will pull populate the unfilled spaces with random stories.

- Constructors

  public FeedController(StoryFeed storyFeed, UserService userService, StoryService storyService)

- Methods

  @RequestMapping(value = "/findFeedForUser/userId", method = RequestMethod.GET)

  public @ResponseBody List<Story>findFeedForUser(@ModelAttribute("user") @PathVariable("userId") int userId)

  @RequestMapping(value = "/findFeedOfUsersStories/userId", method = RequestMethod.GET)

  public @ResponseBody List<Story>findFeedOfUsersStories(@ModelAttribute("user") @PathVariable("userId") int userId)

2 Front-End

The front-end is built using React-Native. The documentation for react native can be found online. I would recommend installing Node.js and then using node package manager to install react-native and all packages. The development environment for react native consists of the following:

- Mac and iOS

  Xcode

  react-native

  Android Studio (API 23)

- Windows/Linux

  Android Studio (API 23)

  react-native

You can install Android API 23 under the settings in Android Studio. For Android you will also need either an emulator (use IP 10.0.2.2 to access localhost), or an android device with at least Android 5.0.0 installed on it.

All components including pages, images, etc. are located in RehabStarFront/src/. Fetch API is used to communicate with the backend. There are some calls already created in RehabStarFront/src/services/MobileService.js and the fetch documentation can be online as well as some examples on Facebook's react-native page.

The splash screen can be found at: RehabStarFront/android/app/main/res. Once in res (resources), in each /mipmap-hdpi directory there will be an icon.png file. If a new splash screen is wanted all that is needed is to save a new icon.png file within the directory. You must use icon.png as the filename.