HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communications Technology

Final Project Group Report

**EcoBikeRental**

Subject: ITSS Software Development

Group 9:

Trần Lê Hoàng – 20176764

Hoàng Tuấn Anh Văn - 20170224

Phạm Văn Khoa - 20176796

Trần Hải Sơn – 20176861

*Hanoi, December 2020*

1. Requirement Analysis

# Introduction

## Objective

This document is to provide information about Eco Bike Rental software.

### Product name: EcoBikeRental software

### Explain:

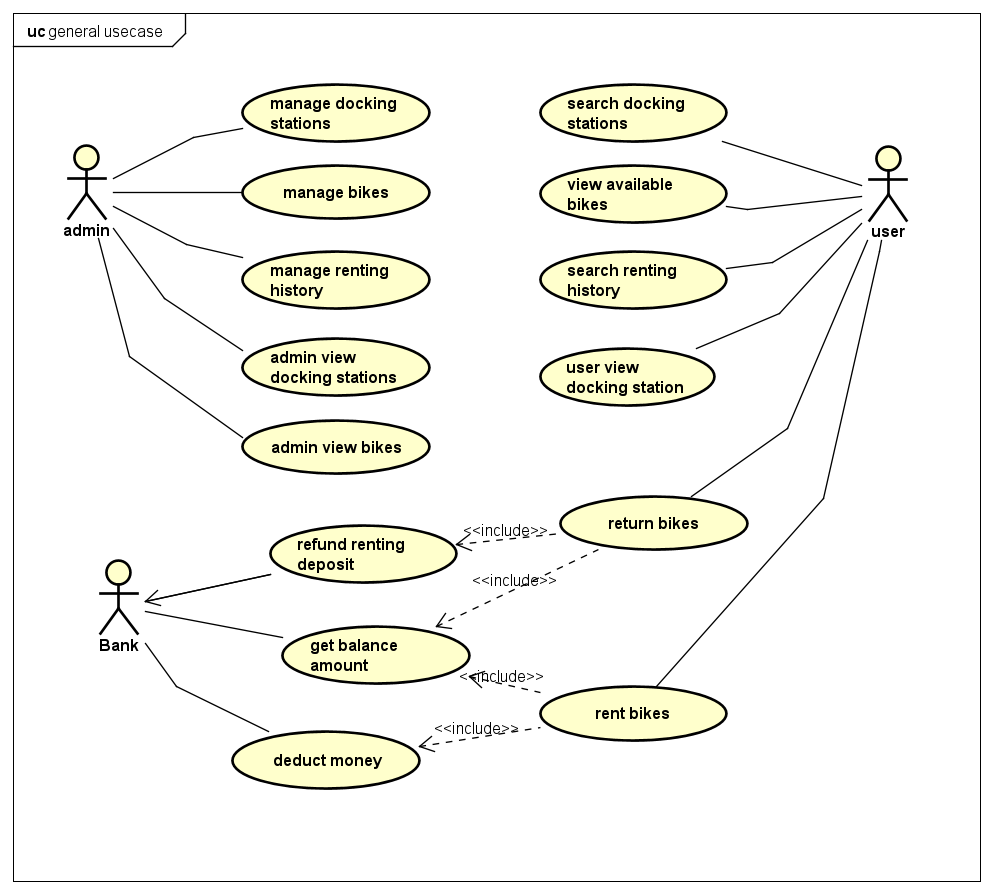
### Software is for users to rent and return bikes automatically. EcoBikeRental is a 24/7 platform-independent system which allows novice users to user without any training. Users must have account to enter to system. Software allow user to enter barcode to rent bike and return bike, use credit card for payment, show information of dock and bike.

### Application:

Software helps to reduce employees, saves money and time. It satisfies needs on bike rental service especially in Ecopark Township. It is expected to serve 100 users at the same time without noticeable loss of performance and to operate in an average of 200 hours without failure. The system also can be repaired within 2 hours after any typical failure. The response time for the system is 1 second or 2 seconds during a peak load if it is not explicitly stated.

# Use case diagram

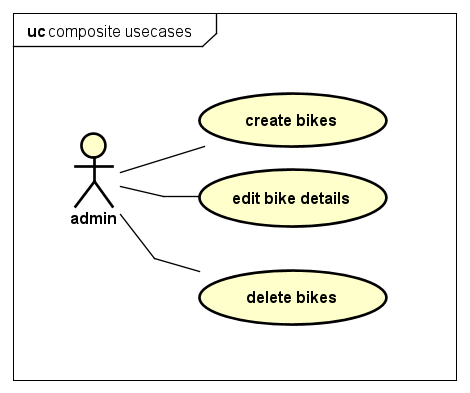
## General use case diagram



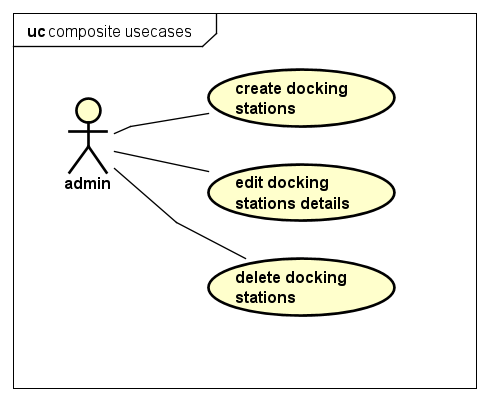
Composite use cases:

* Manage bikes.
* Manage docking stations.
* Manage renting history .
* Rent bikes.
* Return bikes.

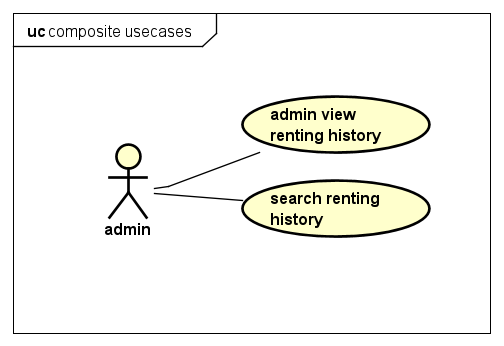
## Use case diagram for “Manage bikes”:



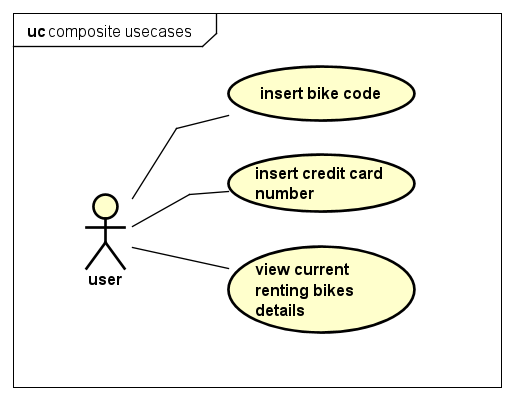
## Use case diagram for “Manage docking stations”:



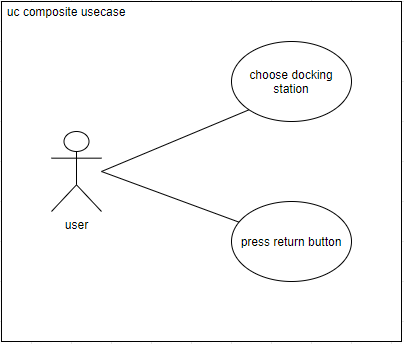
## Use case diagram for “Manage renting history”:



## Use case diagram for “Rent bikes”:



## Use case diagram for “Return bikes”:



# Use case specification

## Use case “Edit bikes detail”

|  |  |  |  |
| --- | --- | --- | --- |
| UC Code | UC001 | Use case name | Edit bike details |
| Actor | Admin | | |
| Description | The “edit bike details” use case is initiated when admin press edit button in the bike detail page. Subsequently, a new page with title “Update bike details” appears containing current information about the selected bike. Admin can then edit each individual fields. The page can be dismissed by clicking either “Save” or “Cancel” button. Pressing “Save” button will result in the entry of selected bike be updated to reflect changes, while “Cancel” button made no changes to that bike | | |
| Precondition | Admin successful Login | | |
| Main flow of event (success) | |  |  |  | | --- | --- | --- | | # | Doer | Action | |  | Admin | Choose edit bike button in bikes page | |  | Software | displays admin edit bike screen | |  | Software | Update details in edit screen with selected bike | |  | Admin | enters bike details to update | |  | Admin | Choose save bike button | |  | Software | checks if the entered detail fields is valid | |  | Software | Save updated bike if details is valid | |  | Software | Displays successfully update bike detail message | |  | Software | Close edit page and return to bike list page | | | |
| Alternative flow of event | |  |  |  | | --- | --- | --- | | # | Doer | Action | |  | Admin | Cancels edit page at any time | | 4a | Admin | Clears any field before saving | | 5b | Admin | Choose cancel button | | 6a. | Software | Notifies error “Invalid field input” if the entered detail field is invalid, go to step 2 | | 6b | Software | Notifies information “No field change” if nothing is edited, go to step 2 | | | |
| Post condition | No | | |

\* Bike edit input requirements:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Data field** | **Description** | **Required?** | **Valid condition** | **Example** |
| 1. | Name | The name of this bike | Yes | Characters and numbers only | Eco Bike |
| 2 | Weight | The bike’s weight (in kilograms) | Yes | Numbers only | 30 |
| 3 | License Plate | The bike’s license series | Yes | Characters and numbers only | ECOB12345 |
| 4 | Manufacturing Date | The bike’s manufacturing date | Yes | Check for valid date format | 12/10/2015 |
| 5 | Producer | The bike’s producer company | Yes | Characters and numbers only | Thong Nhat |
| 6 | Cost | The bike’s cost | Yes | Currency format | 10.000.000 |
| 7 | Docking Station Id | The bike’s station | Yes | Existed station id | Ds1 |

## Use case “Return Bike”

|  |  |  |  |
| --- | --- | --- | --- |
| UC Code | UC003 | Use case name | User Return Bike |
| Actor | User | | |
| Precondition | Rented a bike | | |
| Main flow of event  (success) | |  |  |  | | --- | --- | --- | | # | Doer | Action | |  | User | Press “Return bike” button. | |  | System | Display “Rental Details” dialog. | |  | User | Choose which docking station to return bike. | |  | User | Press “Pay and return” button. | |  | System | Refund deposit money. | |  | System | Deduct renting money. | |  | System | Save new rental. | | | |
| Alternative flow of event | |  |  |  |  |  | | --- | --- | --- | --- | --- | | # | | Doer | | Action | | 3a | System | | Show diaglog “Docking station empty dock left”. | | | 6a | System | | Show dialog “Credit card balance insufficient!”. | | | | |
| Post condition | No | | |

## Use case “Insert Credit Card”

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case code** | UC003 | **Use case name** | Insert credit card |
| **Actor** | User | | |
| **Precondition** | User must be in docking station before renting bike | | |
| **Main flow of event**  **(success)** | |  |  |  | | --- | --- | --- | | No | Doer | Action | | 1. | User | Press "Rent" button. | | 2. | System | Display dialog to enter card number. | | 3. | User | Enter card number (description below \*). | | 4. | User | Request for bike (press confirm button). | | 5. | System | Check if card number is valid or not. | | 6. | System | Check if credit card is available or not. | | 7. | System | Check if credit card has enough money to perform transaction or not. | | 8. | System | Prompt user and move to another screen to show state when renting bike. | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | No | Doer | Action | | 5a. | System | Error: card number is invalid. | | 6a. | System | Error: credit card is being used to rent bike. | | 7a. | System | Error: credit card does not have enough money to perform transaction. | | | |

\* Bike code requirement.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Input | Description | Compulsory? | Valid condition | Example |
| 1. | Card number | User must enter credit card to rent bike, 1 credit card is allowed to rent only 1 bike at a time. | Yes | Card number must be same with valid code in the system | 201768612017 |

## Use case “Search docking stations”

|  |  |  |  |
| --- | --- | --- | --- |
| 1. UC Code | UC004 | Use case name | User Search Docking Stations |
| Actor | User | | |
| Description | The “user search docking stations” use case is initiated when user press the log in as user button in the log in page. Subsequently, a new page with title “Eco Bike Rental for USER” appears containing 3 text fields (ID, Name, Address) for user to enter search keywords. User then can enter some keywords in those fields. System then display a list of station matched the input keywords when user press Search button. | | |
| Precondition | User successful Login | | |
| Main flow of event (success) | |  |  |  | | --- | --- | --- | | # | Doer | Action | | 1 | User | Enter search keywords in search pane search fields | | 2 | User | Press Search Button | | 3 | Software | Get search parameters | | 4 | Software | Search for stations that match the keyword | | 5 | Software | Display list of matched station | | | |
| Alternative flow of event | |  |  |  | | --- | --- | --- | | # | Doer | Action | |  | User | Close Software at any time | | 5a | Software | Display nothing in station list pane if there is no matching station | | | |
| Post-condition | None | | |

\* Input requirement:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Data field** | **Description** | **Required?** | **Valid condition** | **Example** |
| 1 | ID | Keyword for ID of station | No | Characters and digits | 12Bc1 |
| 2 | Name | Keyword for name of station | No | Characters and digits | Station1 |
| 3 | Address | Keyword for address of station | No | Characters and digits | Ta Quang Buu |

# Glossary

## Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal data dictionary, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

## Definitions

The glossary contains the working definitions for the key concepts in the Course Registration System.

### Admin

The person who owns the right to manage various categories in the system including bikes, docking stations,…

### User

The person who interact with the software in order to rent bike from docking station.

### Bike

A bicycle or motorcycle in docking station that users can rent:

* Normal bike: 1 saddle, 1 pedal, 1 behind saddle.
* E-bike: 1 saddle, 1 pedal, 1 behind saddle, motor.
* Twin bike: 2 saddle, 2 pedal, 1 behind saddle.

### Docking Station

An area to rent or return bikes.

### Rental

Information about previous bike renting details.

### Bike id

A unique sequence of characters that is used to specify a bike.

### Credit card

User’s banking credit information that is used to rent or return bike.

### Renting deposit

A sum payable as a first installment on renting a bike (the balance being payable later).

### Balance amount

Amount of money in credit card.

# Supplementary specification

The purpose of this document is to define requirements of the Eco Bike Rental System. This Supplementary Specification lists the requirements that are not readily captured in the use cases of the use-case model. The Supplementary Specifications and the use-case model together capture a complete set of requirements on the system

## Functionality

* In some steps of many use cases, if we have to work with the database and there is an error related to database connecting or database operating, a corresponding message must be displayed in order to distinguish between database-related error and user’s error.
* General displaying format:
* For integer number, comma for thousands separator
* For number, right alignment
* For message, left alignment
* White background

## Usability

* Functions and features are designed to optimize user’s experience and can be operated simply.
* The novice user should not need to be trained in order to use the software.
* Need a detailed guide for user’s error so that he/she knows how to navigate when an error is met

## Performance

* The software should always operate correctly, responsively in any general cases. In some special cases, a slight drop in performance, response time is allowable.
* Implicitly stated, ideally, the response time for any tasks, with a moderate load, within the system is 1 second. But in case of peak load, a response time in the interval of 2 seconds is admissible.

## Reliability

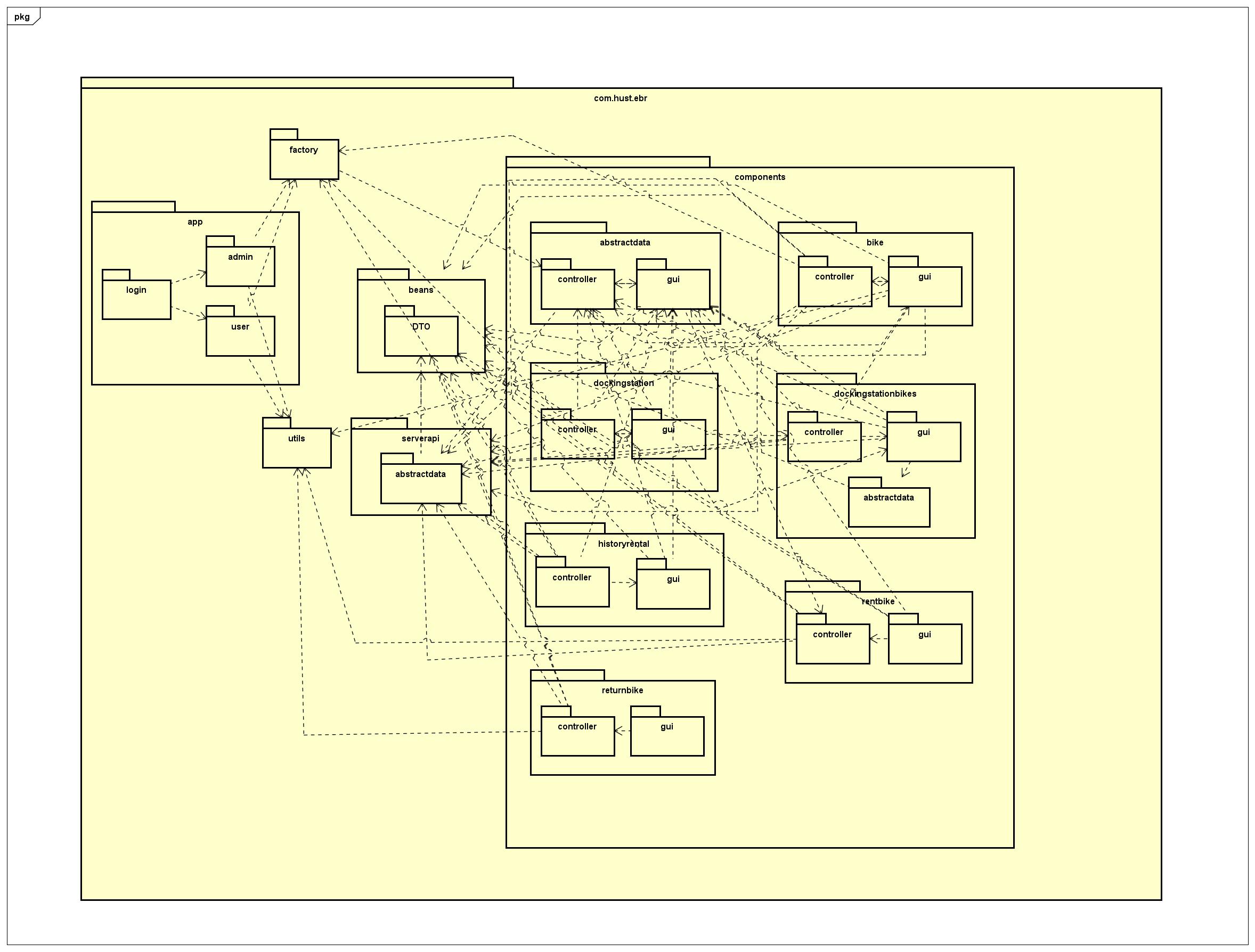
* The system should be able to serve a good number of users in discrete time spans.
* In the case of simultaneous users, the system is expected to serve up to 100 requests.
* The system should run smoothly, consecutively, automatically and reliably. Ideally, a time span of more than 200 hours operating without failure is acceptable.

## Security

The system separates users and admin functionality, users cannot log in as admin to modify stations and bikes.

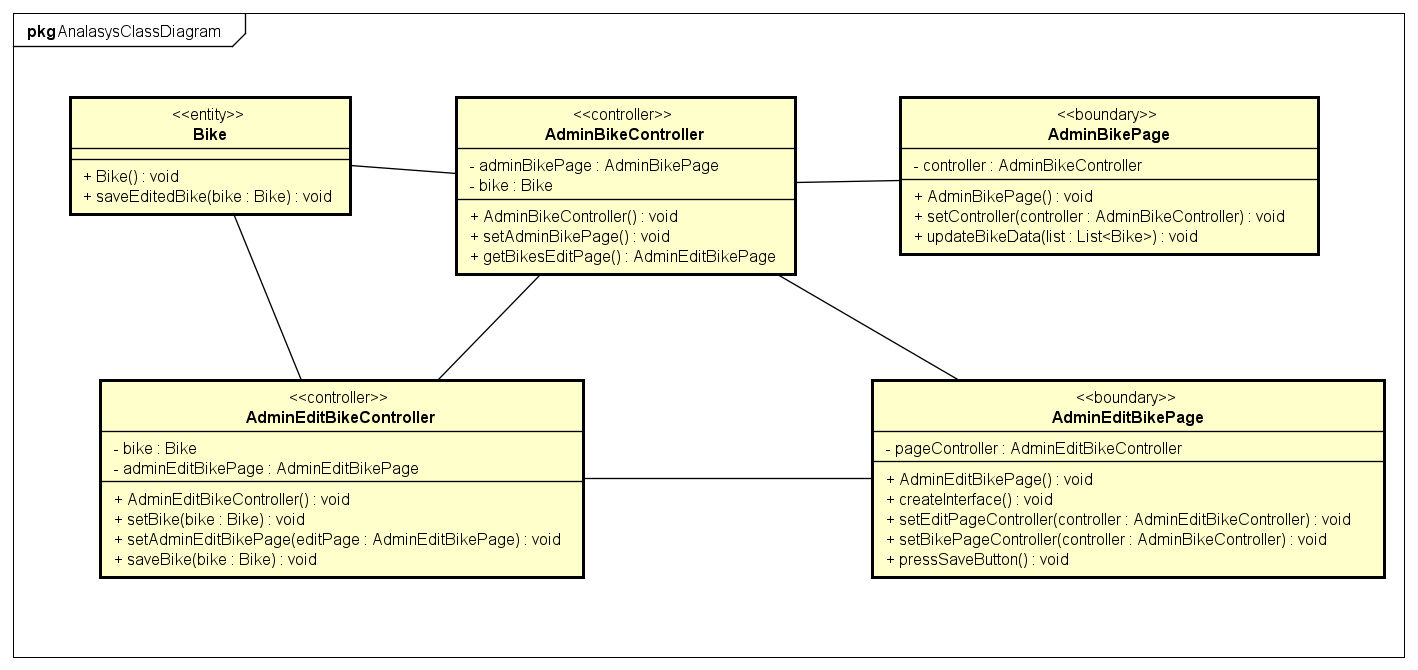
1. Application Design

# Packages diagram

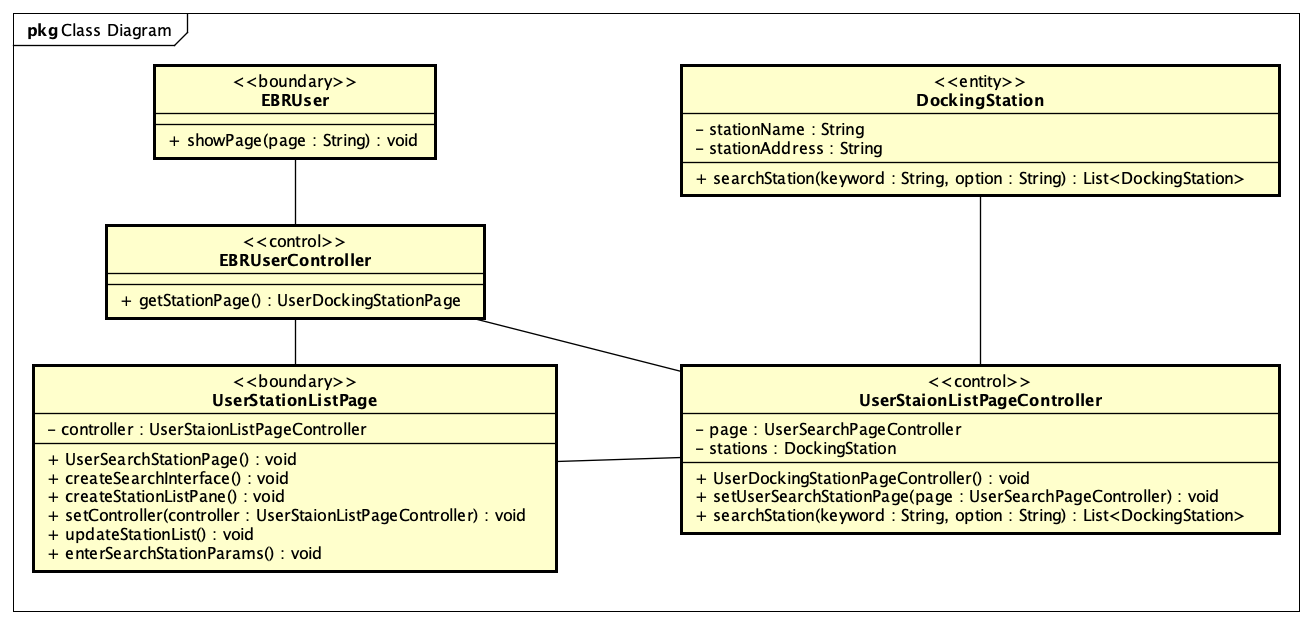


# Analysis class diagram

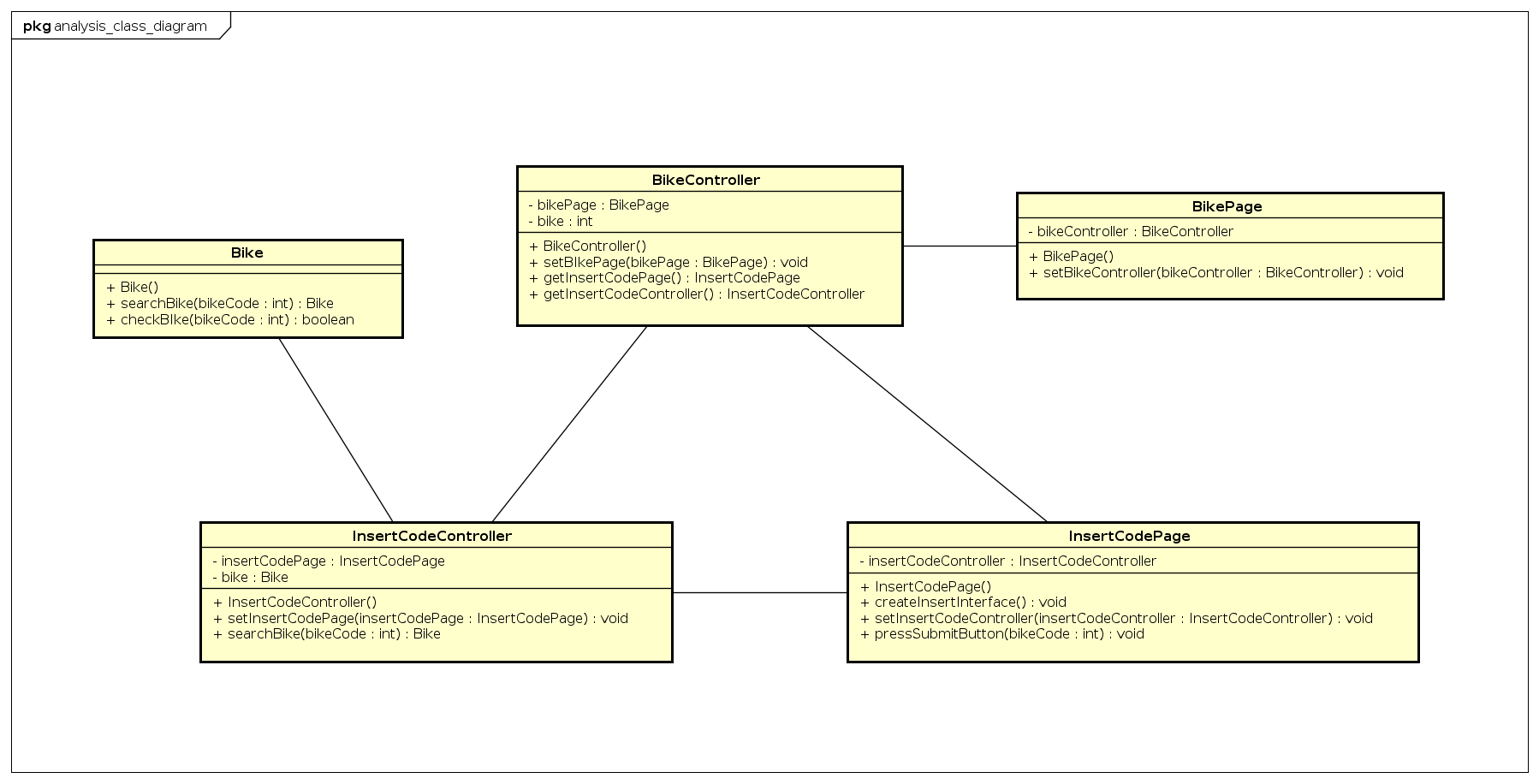
Admin edit bike:



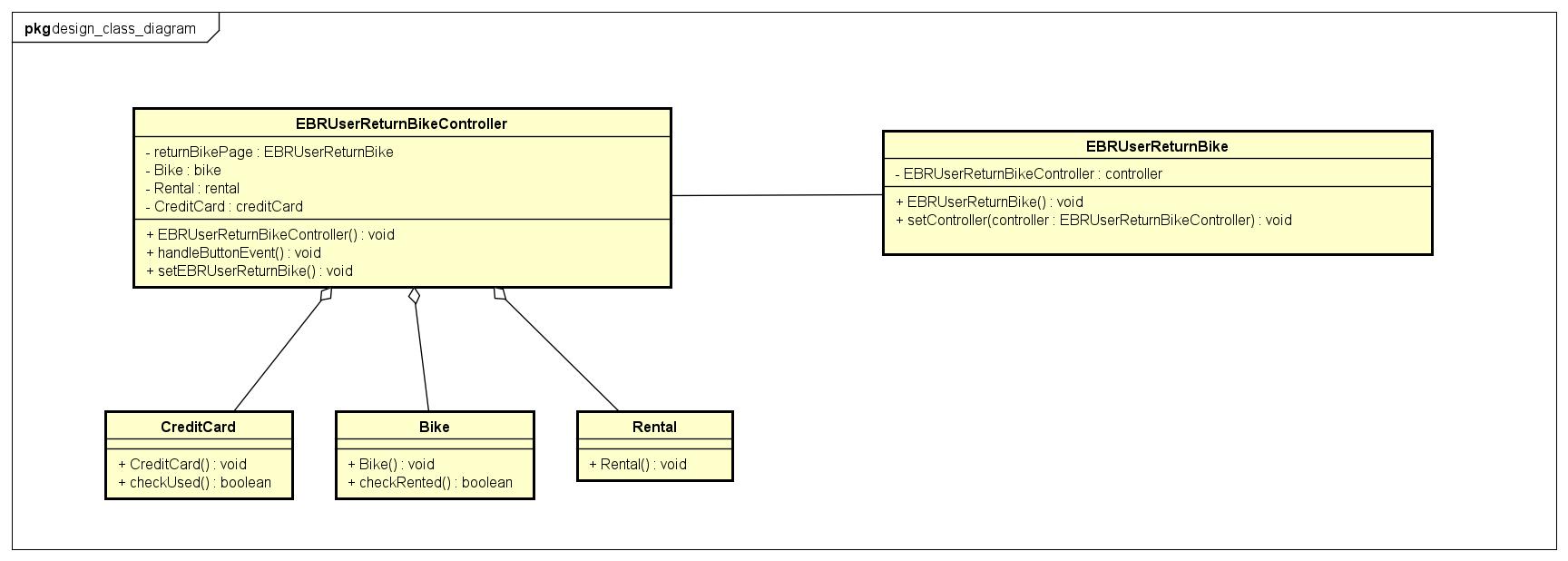
User search docking station:



User insert bike code:

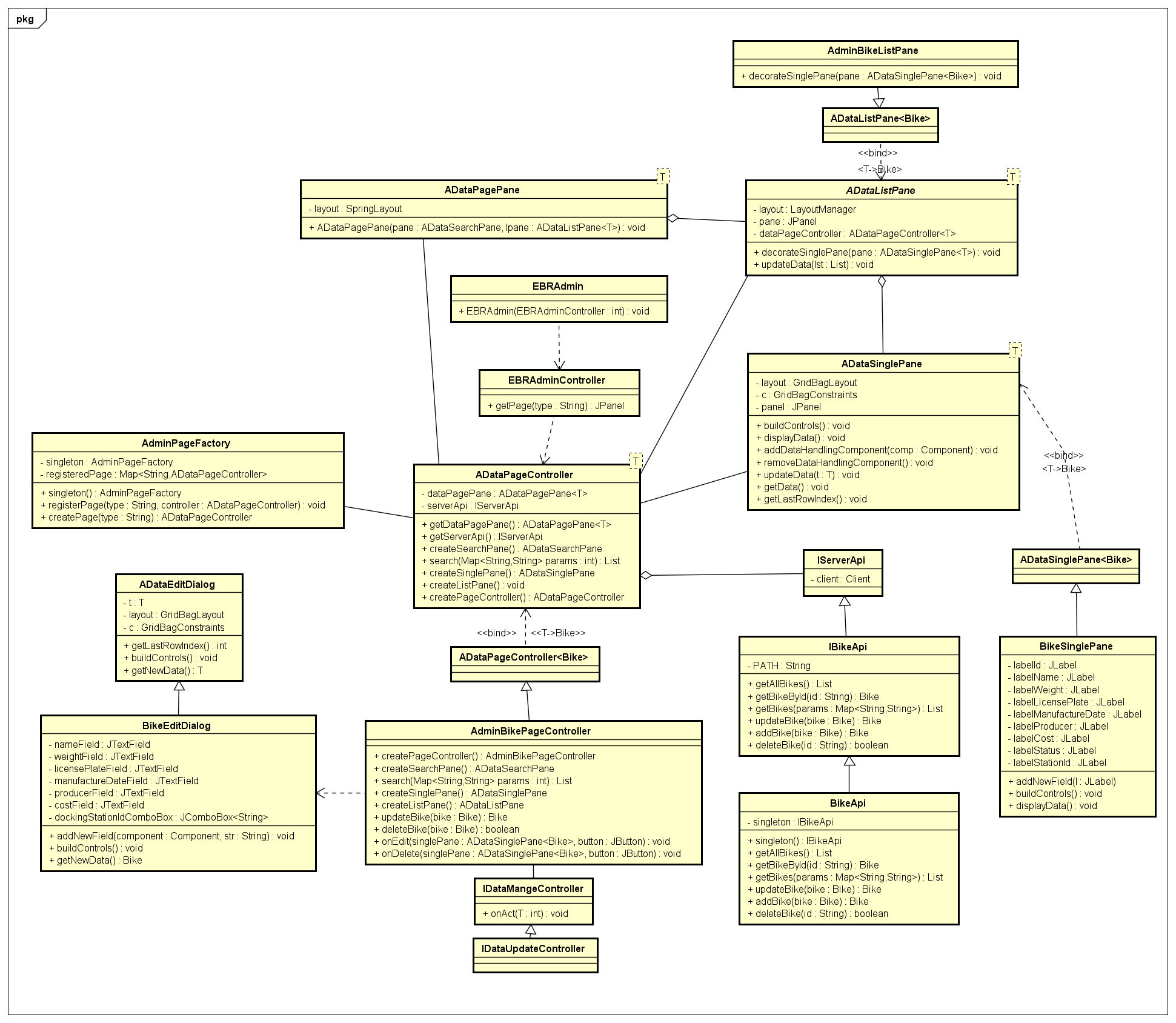


User return bike:

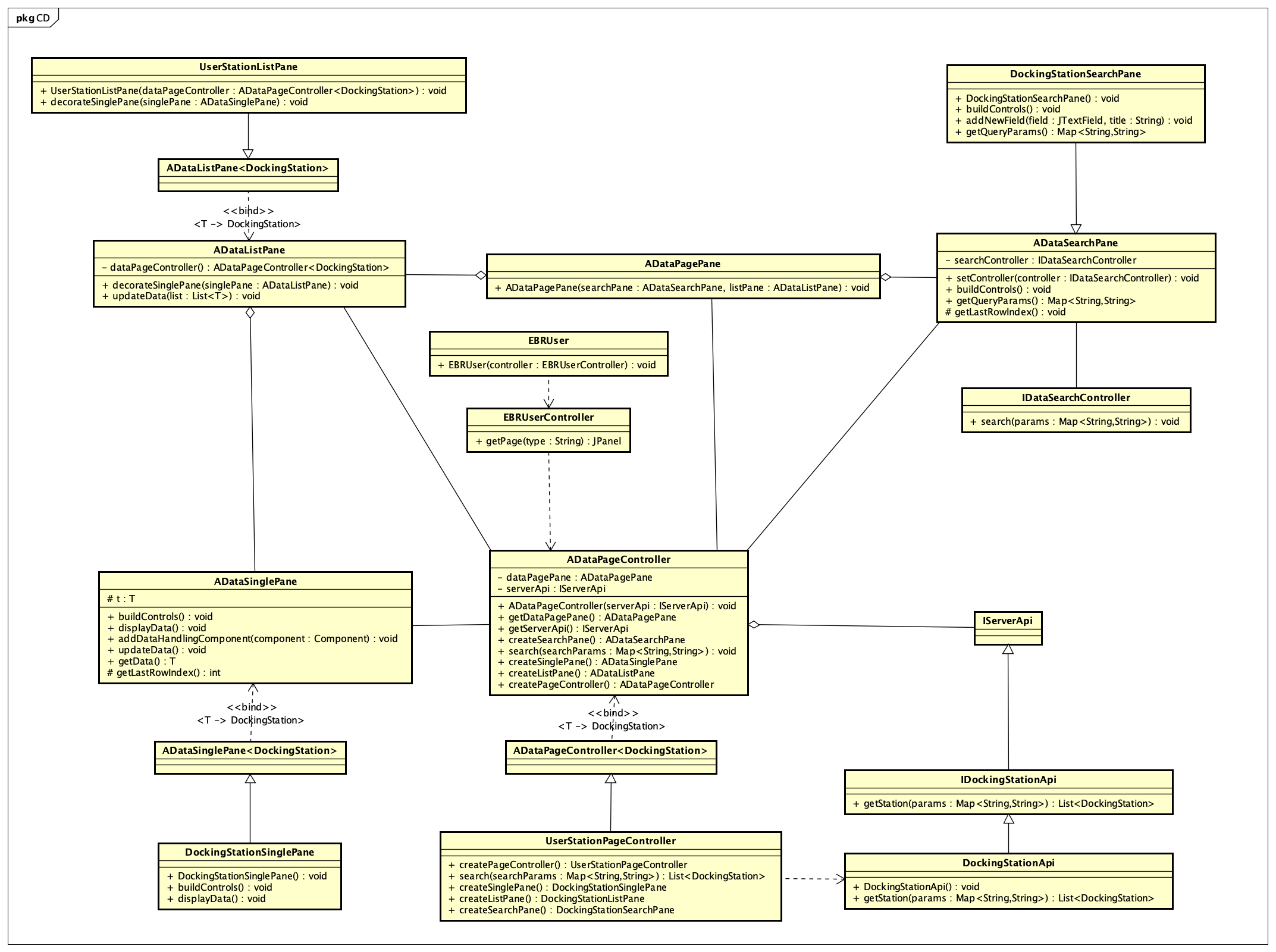


# Design class diagram

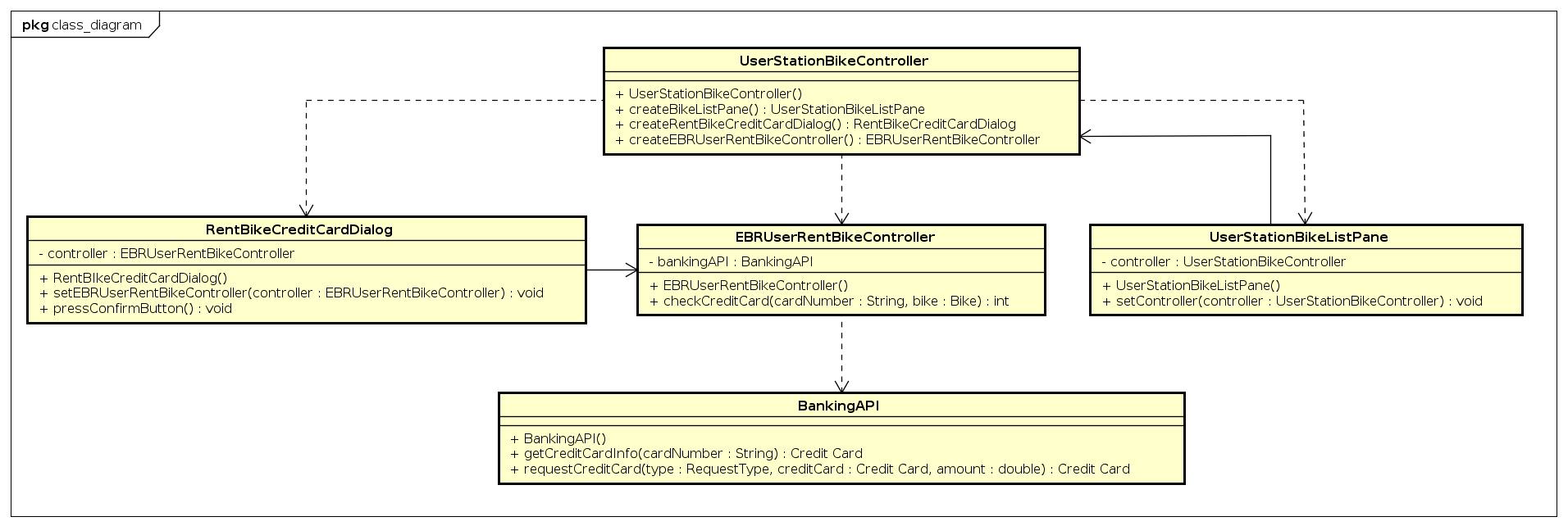
Admin edit bike:



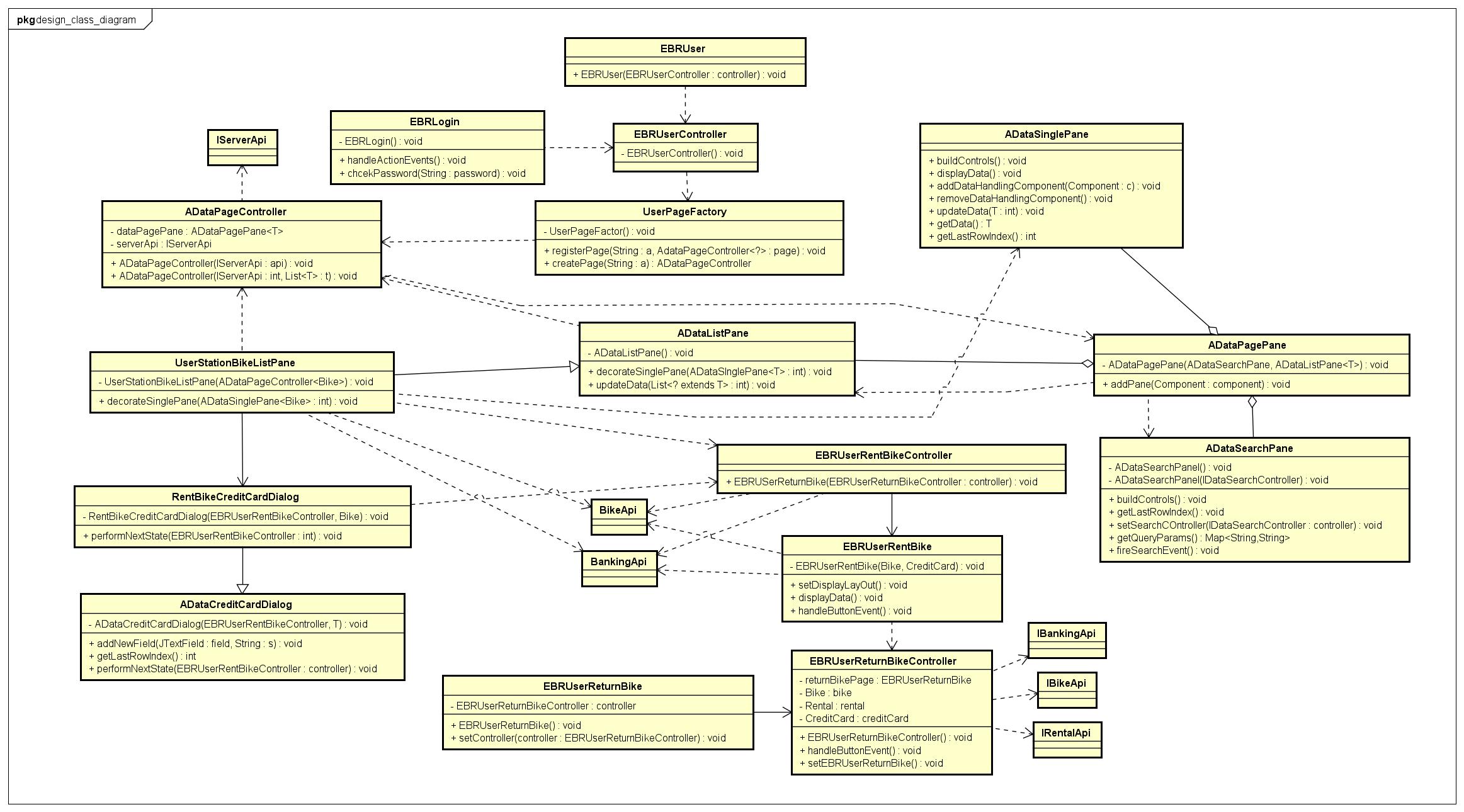
User search docking station:



User insert bike code:



User return bike:



1. Group working diary

Total time allowed to work and finish building the EcoBikeRental software application: 6 weeks.

For each week, each member in the group worked in their specific use case.

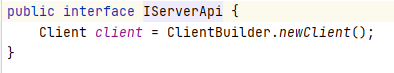
* **First week:** Pick an use case, draw use case diagrams and make specific description on the use case.
* **Tran Le Hoang:** Hoang chose use case “Admin edit bike”, drew use case diagrams, specifically described the use case and completed the SRS file.
* **Pham Van Khoa:** Khoa chose use case “User return bike”, drew use case diagrams, specifically described the use case and completed the SRS file.
* **Hoang Tuan Anh Van:** Van chose use case “User search docking station”, drew use case diagrams, specifically described the use case and completed the SRS file.
* **Tran Hai Son:** Son chose use case “Insert bike code”, drew use case diagrams, specifically described the use case and completed the SRS file.
* **Second week:** Design the architecture and analyze use cases.
* **Tran Le Hoang**: Drew analysis sequence diagram and analysis class diagram of “Admin edit bike” use case.
* **Pham Van Khoa:** Drew analysis sequence diagram and analysis class diagram of “User return bike” use case.
* **Hoang Tuan Anh Van:** Drew analysis sequence diagram and analysis class diagram of “User search docking station” use case.
* **Tran Hai Son:** Drew analysis sequence diagram and analysis class diagram of “Insert bike code” use case.
* **Third week:** Design UI.
* **Group:** Drew screen transition diagrams.
* **Tran Le Hoang:** Specifically describe “Admin edit bike” screens.
* **Pham Van Khoa:** Specifically describe “User return bike” screens.
* **Hoang Tuan Anh Van:** Specifically describe “User search docking station” screens.
* **Tran Hai Son:** Specifically describe “Insert bike code” screens.
* **Fourth week:** Design classes in detail.
* **Group:** From every class diagrams of each member, logically organized classes into packages.
* **Tran Le Hoang:** Drew sequence diagram and class diagram based on “Admin edit bike” designed classes.
* **Pham Van Khoa:** Drew sequence diagram and class diagram based on “User return bike” designed classes.
* **Hoang Tuan Anh Van:** Drew sequence diagram and class diagram based on “User search docking station” designed classes.
* **Tran Hai Son:** Drew sequence diagram and class diagram based on “Insert bike code” designed classes.
* **Fifth week:** Programming and Testing
* **Tran Le Hoang:** Set up server and json files, worked in programming use case “Admin edit bike” including GUI and Controller, server API, beans package, add/delete bikes and later refactor.
* **Pham Van Khoa:** Worked in programming use case “User return bike” including GUI and Controller, functions to calculate renting money, max amount of time to rent, and new function 24hRentingCost, reformed rent/return bike GUI.
* **Hoang Tuan Anh Van:** Worked in programming use case “User search docking station” including GUI and Controller, HomePage, add/delete docking station, search bike in docking station, testAPI.
* **Tran Hai Son:** Worked in programming use case “User rent bike” including GUI and Controller, InsertCreditCard dialog, checkCreditCard.
* **Sixth week:** Final edit and refactor, merge all works, preparing necessary files for final presentation.

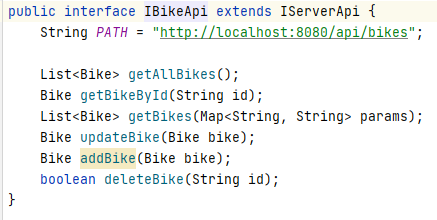
1. Refactor
2. Dependency Injection principle, singleton design pattern, interface segregation, single responsibility for ServerApi

- Initially, AdminBikePageController in upper level depends on ServerApi in lower level. Therefore interface IServerApi was created to resolve the problem.

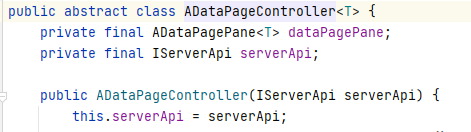
- Additionally, in order for ServerApi scalability, Interface segregation and single responsibility principles are applied to separate interface IServerApi into IBikeApi, IDockingStationApi, IBankingApi, IRentalApi which are responsible for corresponding api tasks.

- Each server api may get initialized multiple times, hence single design pattern is applied for optimization.





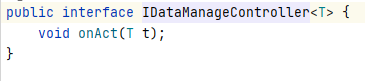




1. Interface segregation for IDataManageController

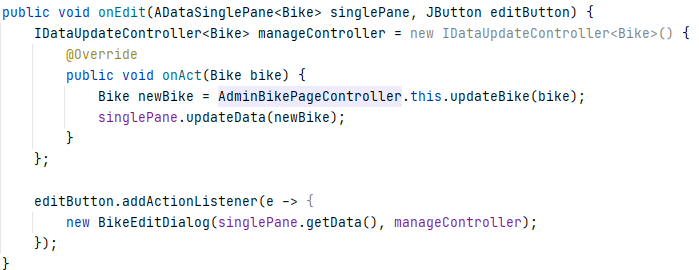
- Initially, interface IDataManageController contains 4 data managing methods CRUD. However some classes such as AdminBikePageController or AdminStationPageController only implement the update method of this interface, therefore other methods are redundant. In order to resolve the problem, IDataManageController is separated into IDataUpdateController, IDataCreateController, IDataDeleteController, IDataReadController





1. Single responsibility for AdminBikeListPane, AdminStationListPane

-> AdminBikeListPane, AdminStationListPane do not ensure MVC pattern and singleresponsibility since decoratesinglepane should only handle UI, but not init IDataManageController.



1. Factory method for creating page controller

- UserPageFactory and AdminPageFactory classes are created to overcome the limitation of having to modify both EBRAdminController and EBRAdmin (or User) every time a new tab is register into the software.

By implementing factory methods, class will have createPageController(type: String) to create a new controller.



