

Bike portal
Analysis and Design Document
Student: Ács Dávid
Group: 30432

	Version: 0.2
Analysis and design document	Date: 04/Apr /18

Revision History

Date	Version	Description	Author
18/Mar/18	0.1	Housekeeping	Ács Dávid
04/Apr/18	0.2	Architecture and views	Ács Dávid

	Version: 0.2
Analysis and design document	Date: 04/Apr /18

Table of Contents

I.	Project Specification	4
II.	Elaboration – Iteration 1.1	4
1.	Domain Model	4
2.	Architectural Design	5
2.1	Conceptual Architecture	5
2.2	Package Design	6
2.3	Component and Deployment Diagrams	7
III.	Elaboration – Iteration 1.2	8
1.	Design Model	8
1.1	Dynamic Behavior	8
1.2	Class Design	9
2.	Data Model	10
3.	Unit Testing	10
IV.	Elaboration – Iteration 2	11
1.	Architectural Design Refinement	11
2.	Design Model Refinement	11
V.	Construction and Transition	11
1.	System Testing	11
2.	Future improvements	11
VI.	Bibliography	11

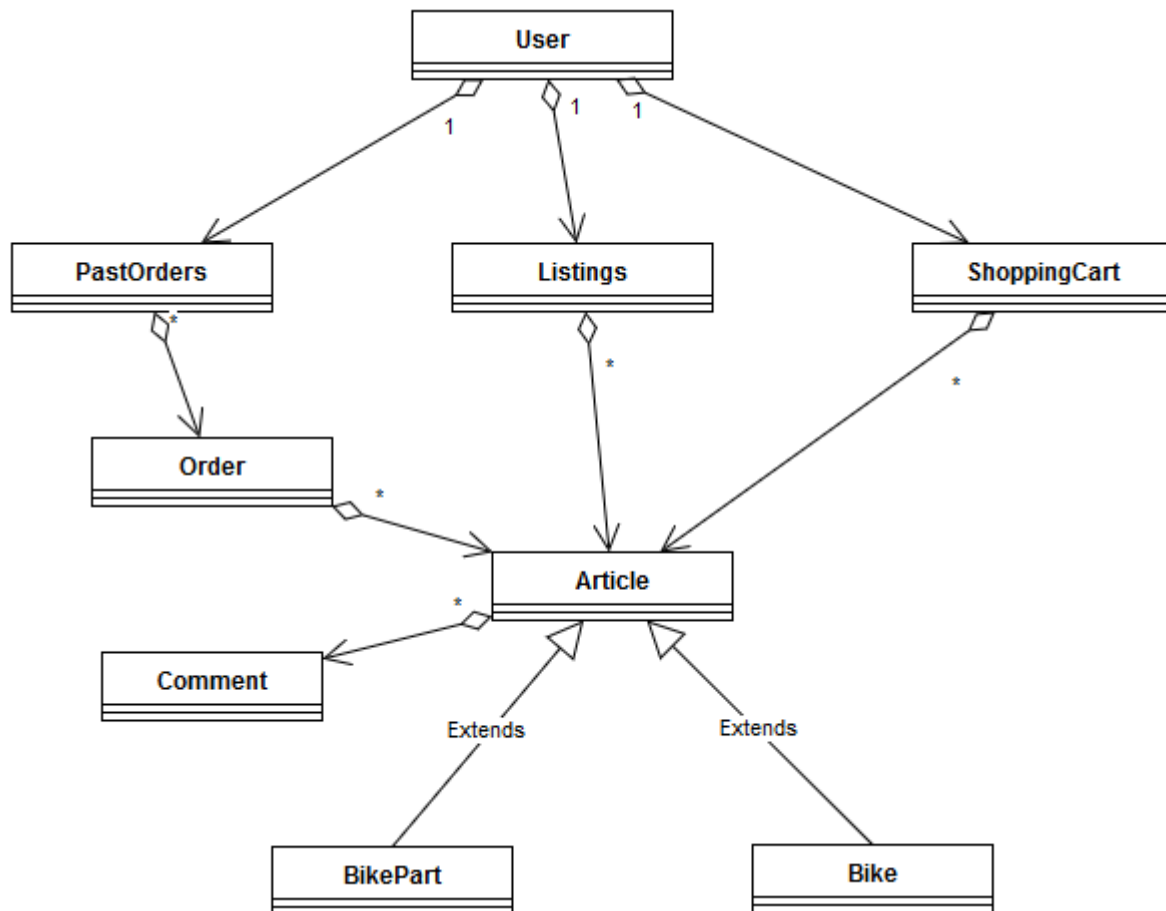
	Version: 0.2
Analysis and design document	Date: 04/Apr /18

I. Project Specification

Bike portal is an online Bike and bike parts store. User can buy/sell bikes and related parts online. To buy products, the user must be registered and logged in, and should add the products to the shopping cart. To check out the user can pay by credit card. Users may rent bikes, and the admin may approve these requests. The admin of the portal can delete, modify announcements/listing if they are inappropriate.

II. Elaboration – Iteration 1.1

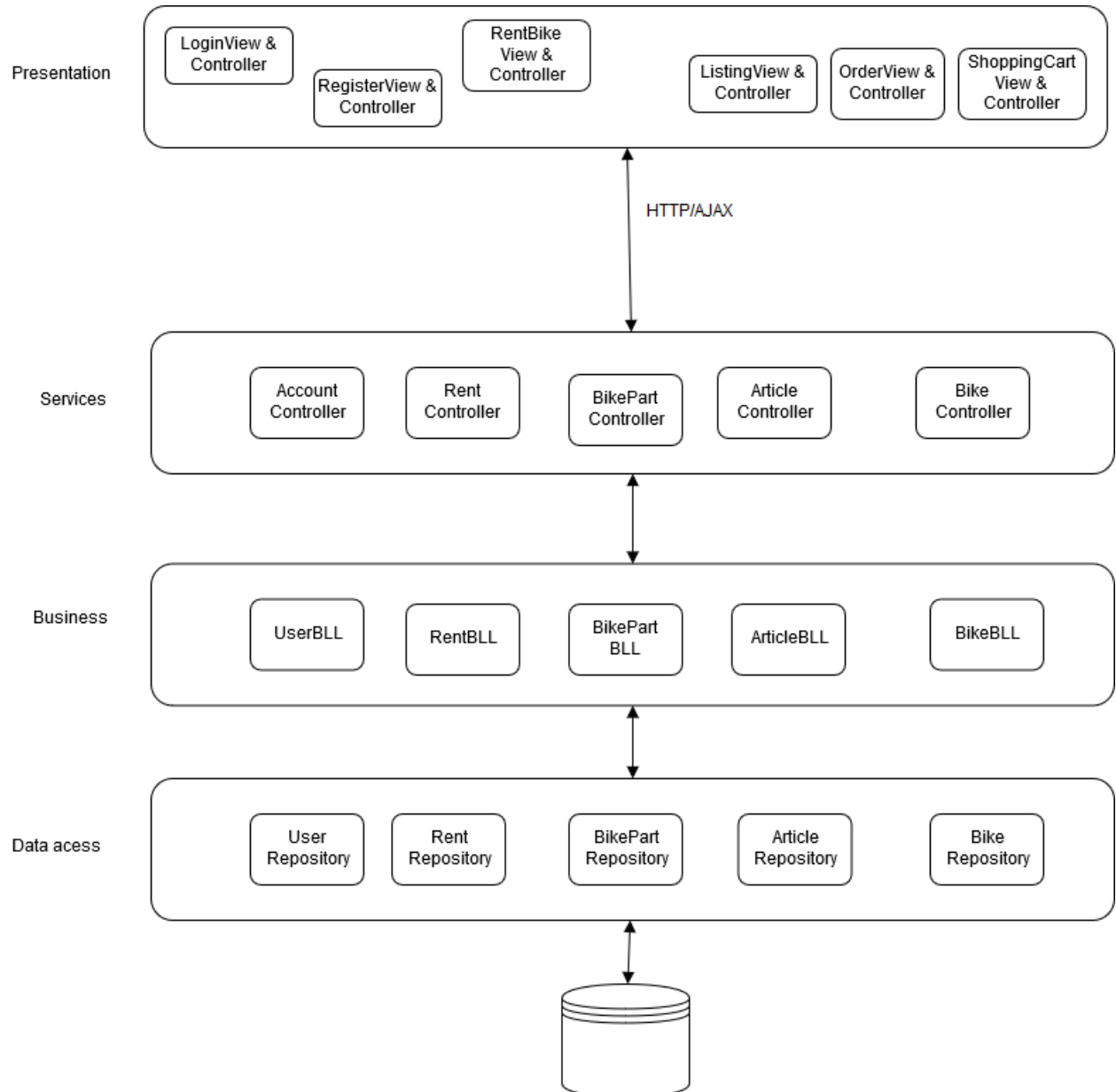
1. Domain Model



	Version: 0.2
Analysis and design document	Date: 04/Apr /18

2. Architectural Design

2.1 Conceptual Architecture

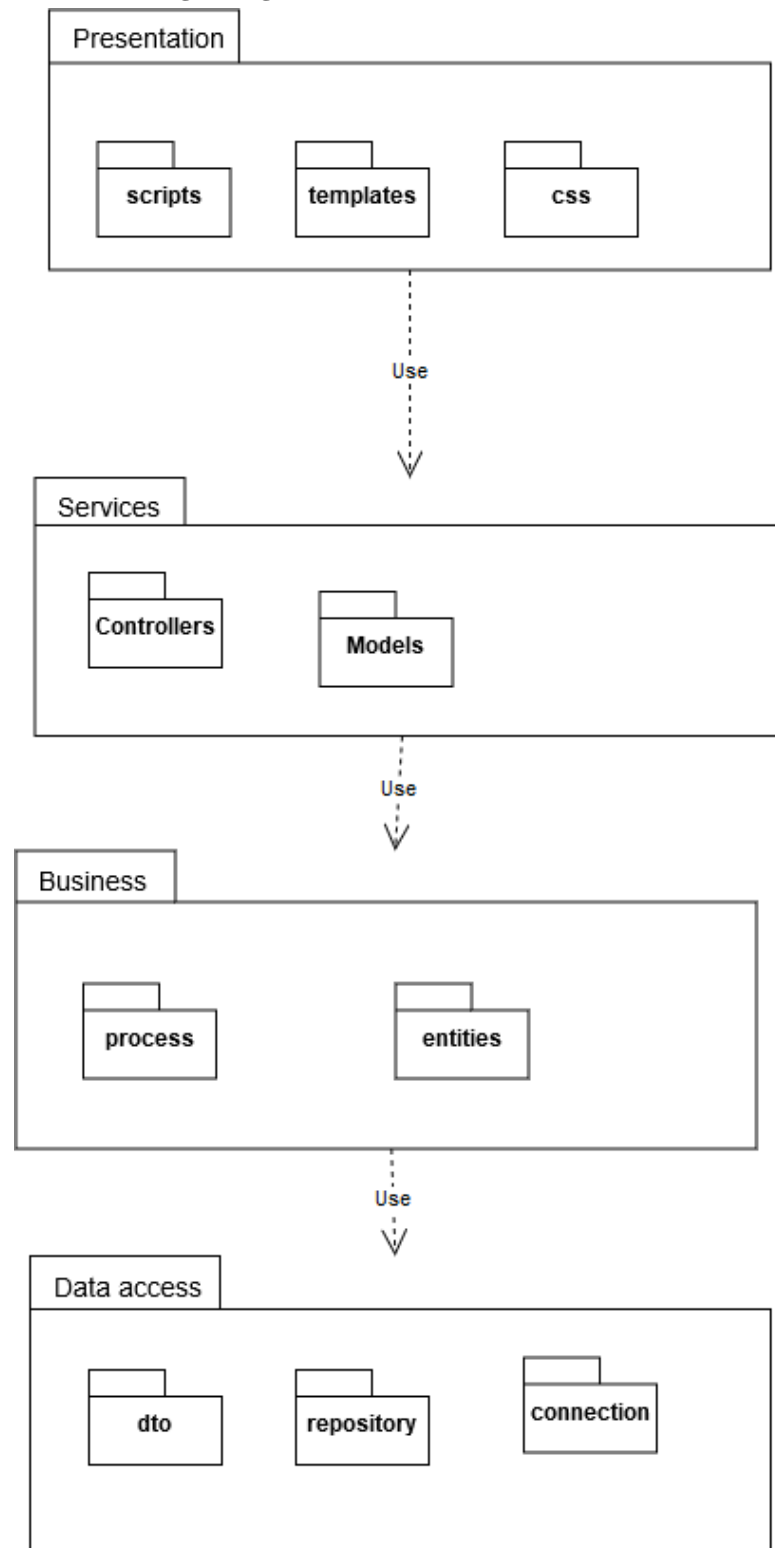


Client server architecture is used in this architecture, to separate the server-side application (data and logic) from the client-side (UI). This way the client-side and the server side may evolve independently of each other.

The separation can be observed, where the HTTP/AJAX calls are made over the network.

	Version: 0.2
Analysis and design document	Date: 04/Apr /18

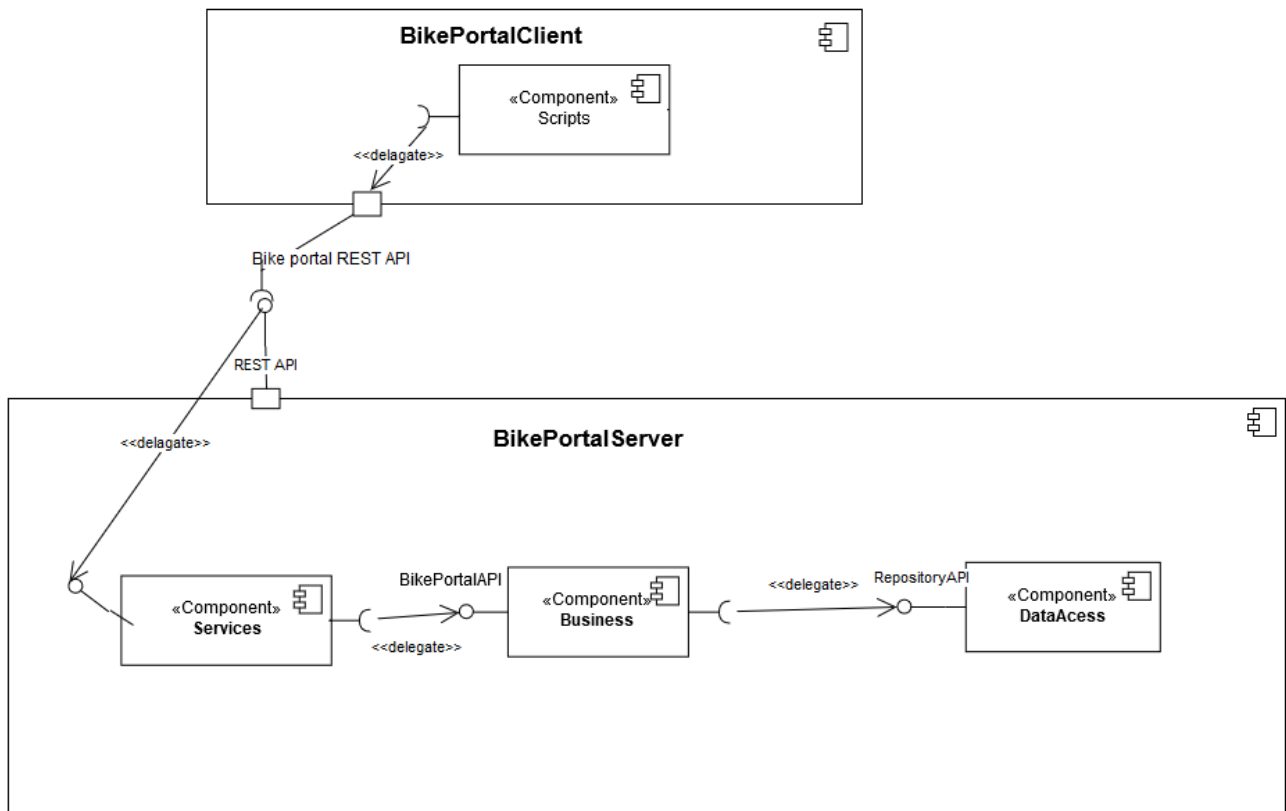
2.2 Package Design



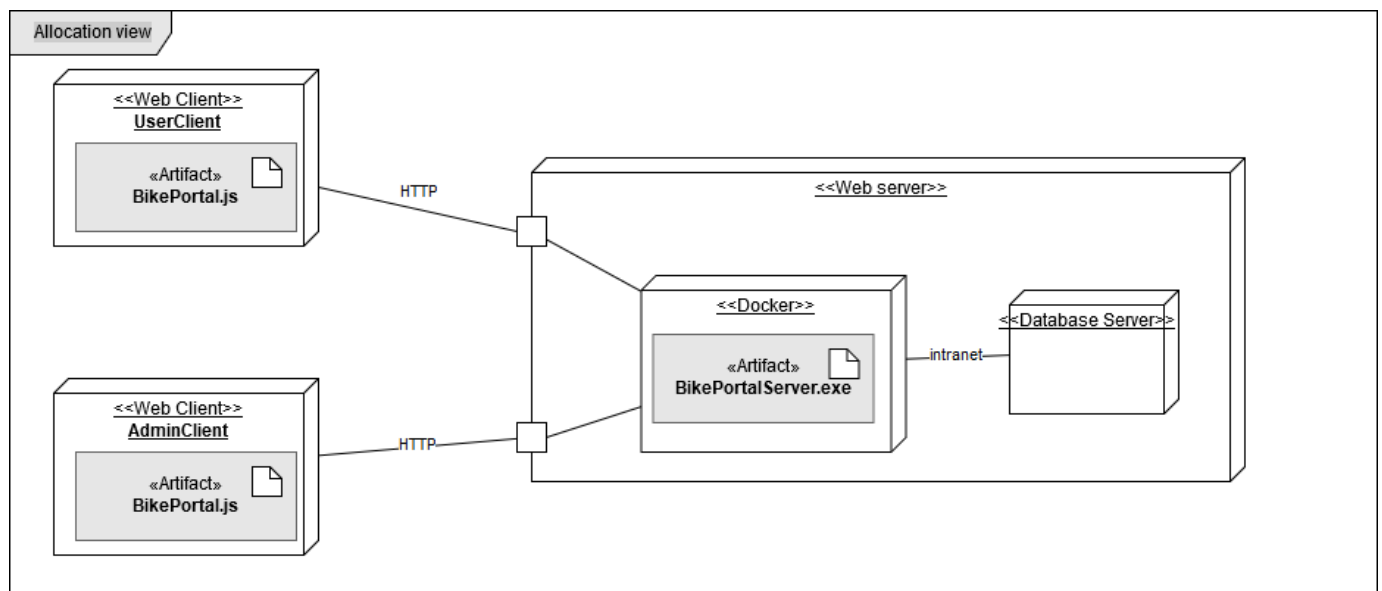
The package diagram of the system represents the layered structure of the application.

	Version: 0.2
Analysis and design document	Date: 04/Apr /18

2.3 Component and Deployment Diagrams



The component diagram shows the clear delimitation between the server and the client. They can only communicate through HTTP, or as it is shown in the diagram, through the REST API.



In the allocation view we may notice that there may be more than one client, and they role may differ depending on who is logged in.

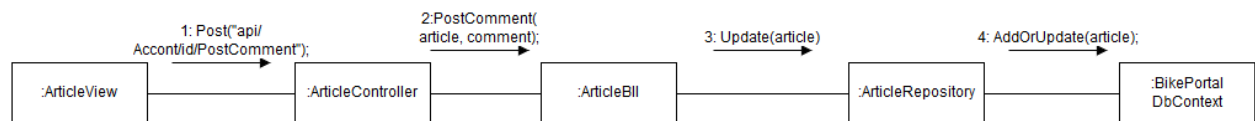
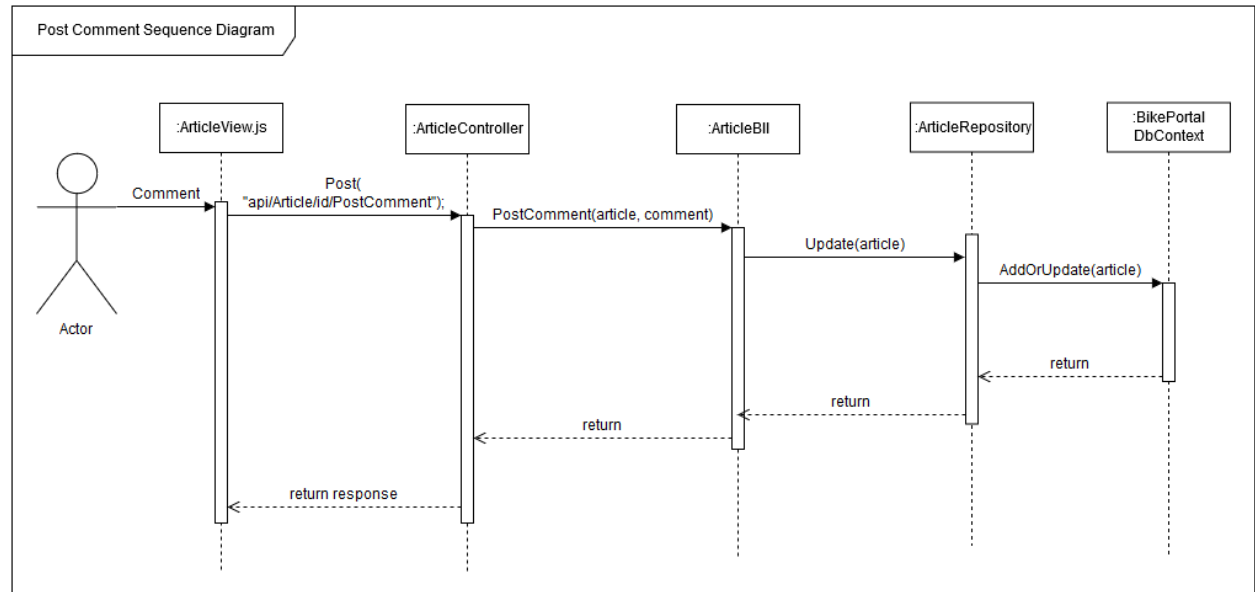
	Version: 0.2
Analysis and design document	Date: 04/Apr /18

III. Elaboration – Iteration 1.2

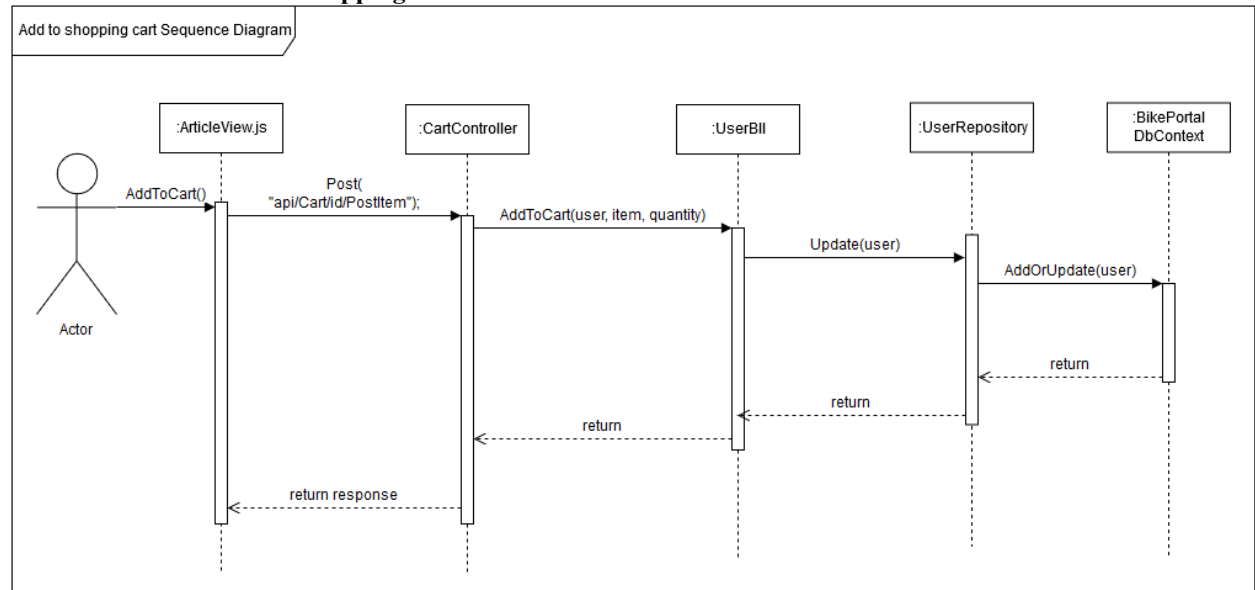
1. Design Model

1.1 Dynamic Behavior

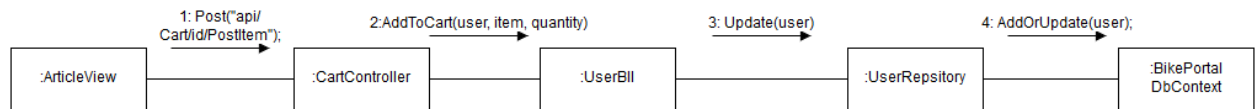
Post a comment to an article.



Add an item to the user's shopping cart.

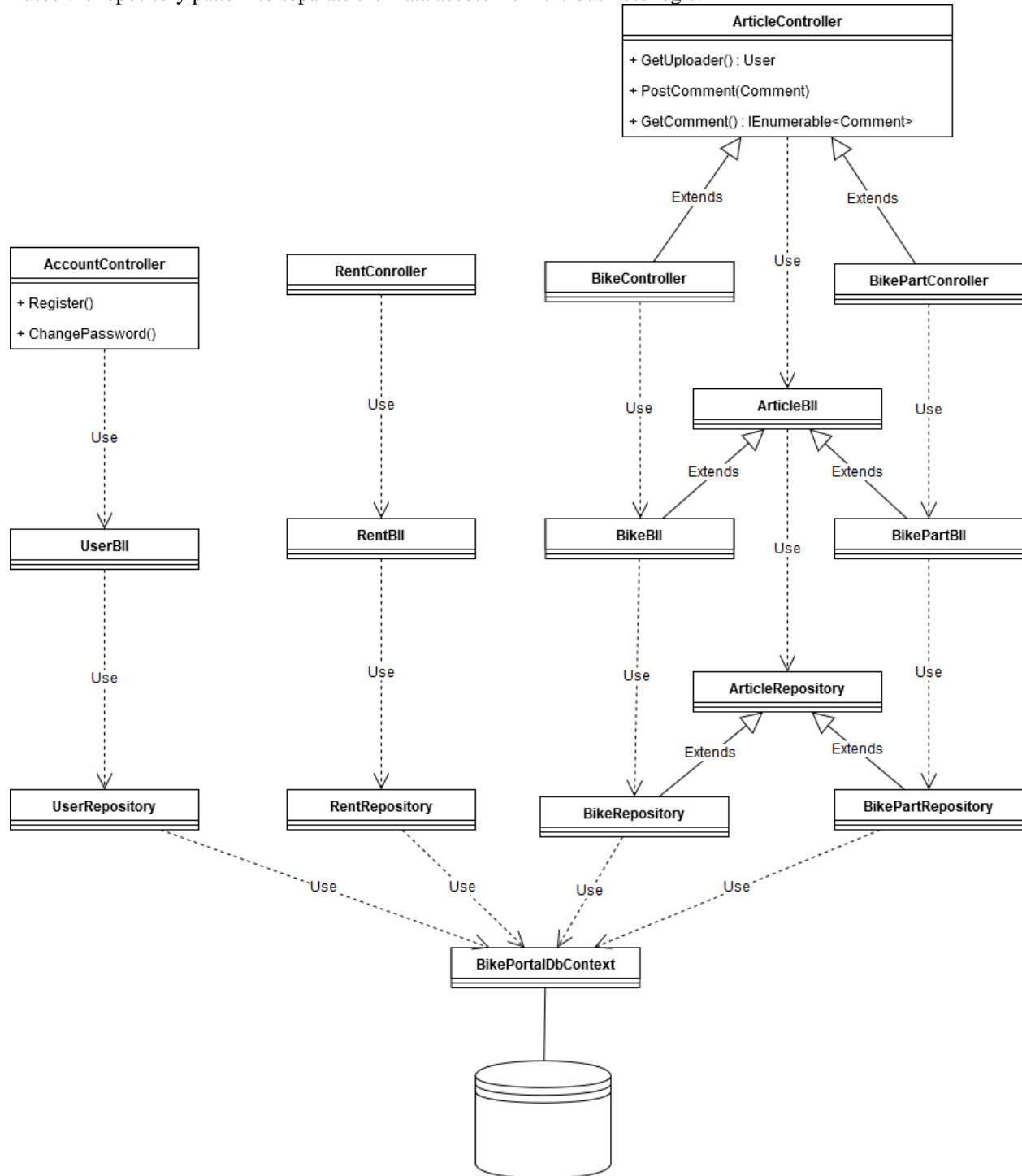


	Version: 0.2
Analysis and design document	Date: 04/Apr /18



1.2 Class Design

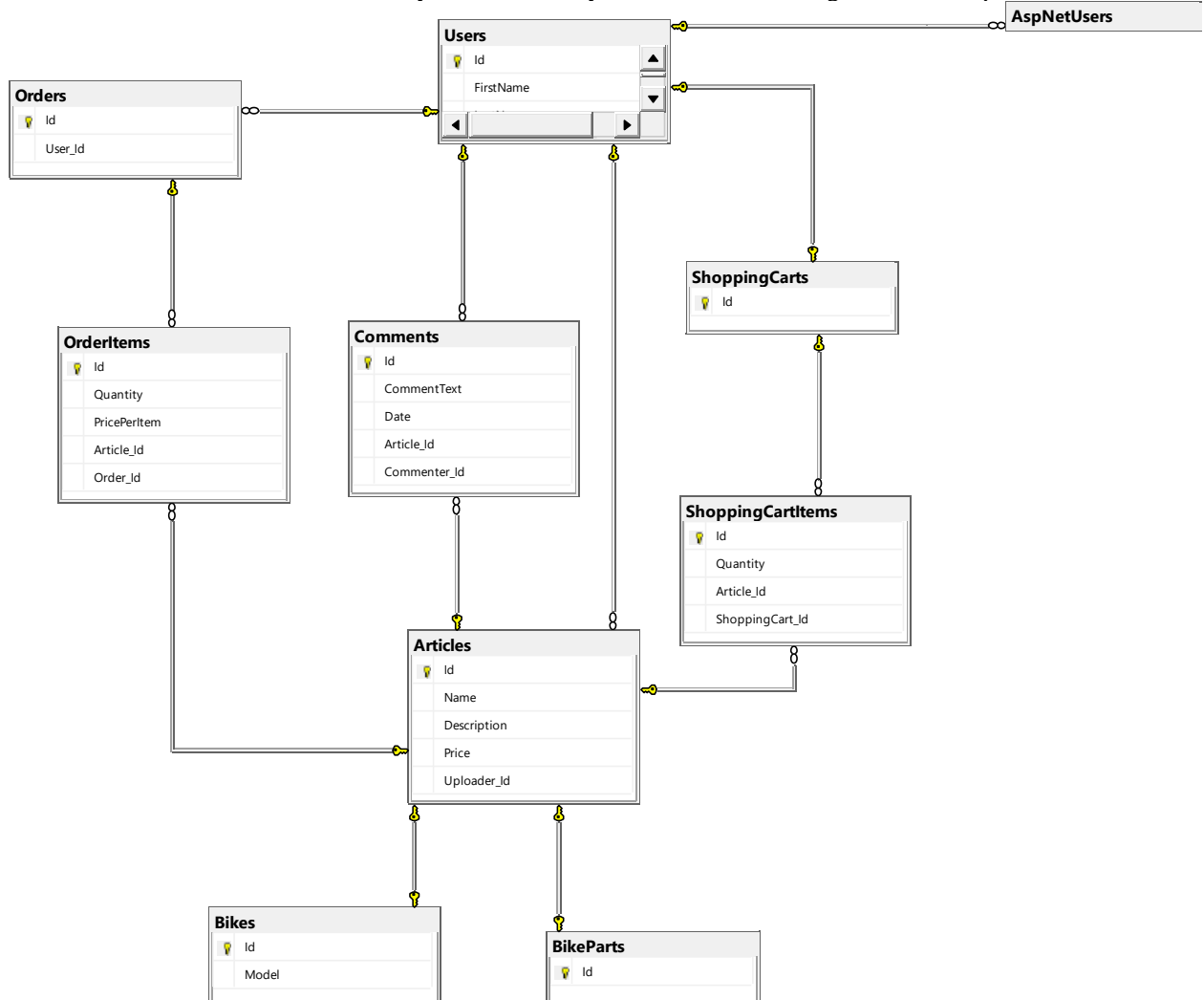
I used the repository pattern to separate the Data access from the business logic.



	Version: 0.2
Analysis and design document	Date: 04/Apr /18

2. Data Model

The data model closely resembles the domain model. In order to create the data model, first I designed domain classes and let the framework create the necessary tables. Bellow you can see the resulting tables of this process:



3. Unit Testing

For unit testing mocking will be used to separate dependencies from the tested component.

Unit Test scenarios

1. Check that the user can login.
2. Check that users can view articles regardless if they are logged in or not.
3. Check that users can compare two items.
4. Check that users can post comments on articles if they are logged in.
5. Check that users can add articles to their shopping carts.
6. Check that users can buy the items already in their shopping carts.
7. Check that users can put a bike as a listing (sell their bike).
8. Check that users can create a request to rest a bike.
9. Check that administrators can delete inappropriate listings from the website.

	Version: 0.2
Analysis and design document	Date: 04/Apr /18

10. Check that administrators can approve requests to rent a bike.

Integration testing will be performed as well to ensure that the individual components fit together in an orderly manner.

System testing will be performed last, to ensure that the system will satisfy all functions and non-functional requirements of the project.

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]

VI. Bibliography