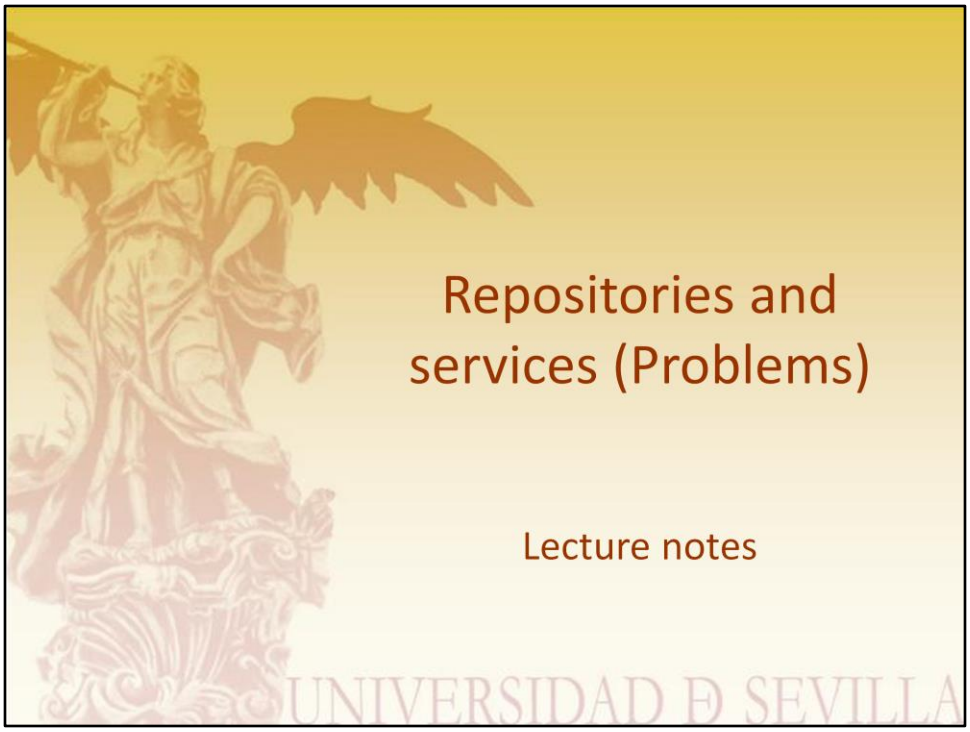


Welcome to this lecture! Today, we'll start putting our knowledge on repositories and services to practice.

--

The problem: a bulletin board

UNIVERSIDAD Ð SEVILLA

We'll start with quite a simple project: a bulleting board to which people can freely post bulletins.

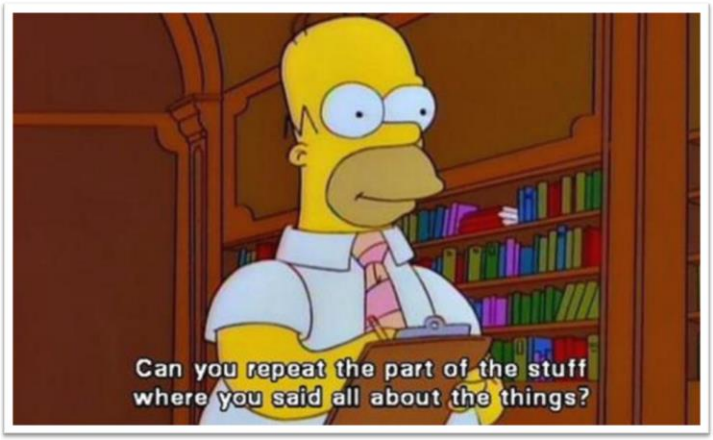
It’s like riding this bike because we’ve provided you with a lot of support. The materials that accompany this lecture provide a project called “Bulletin-Board” and it’s almost complete; you only need to provide a repository and a service for it to work. Please, note that we won’t provide a detailed description of what you have to do to load the project or to set the database up since we assume that you already command these basic steps.

You're constrained!

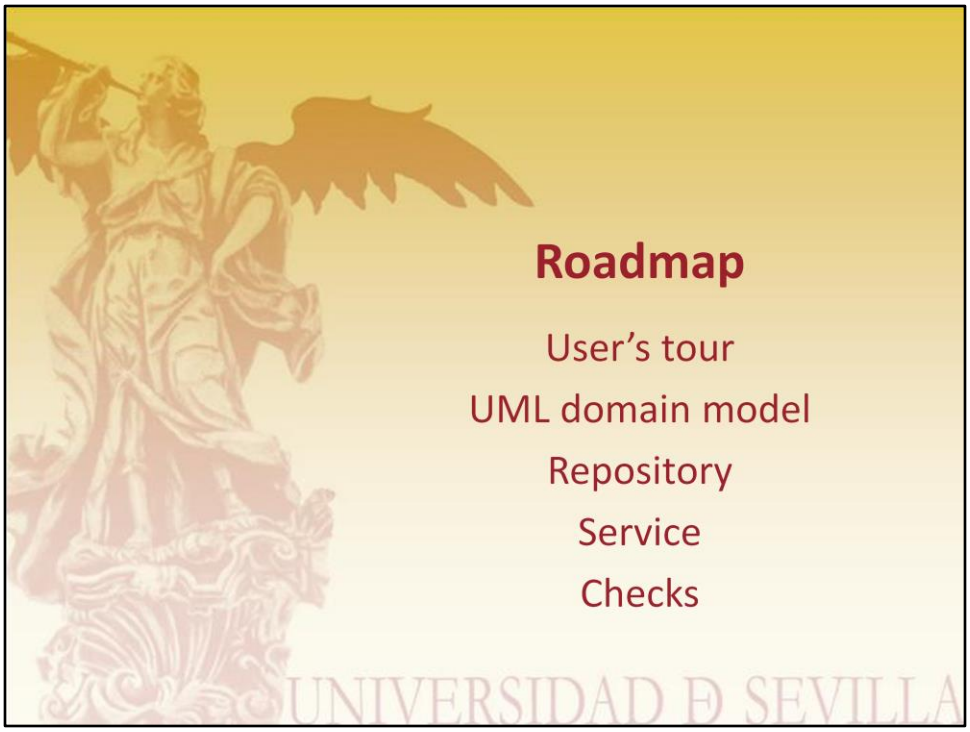
UNIVERSIDAD Ð SEVILLA

Please, note that you're totally constrained. Your only duty in this project is to implement a repository and a service; you aren't allowed to change anything else. Please, take this very seriously: the other parts of the project were implemented by other people; you can't change the artefacts they've produced and hope this doesn't mess things up!

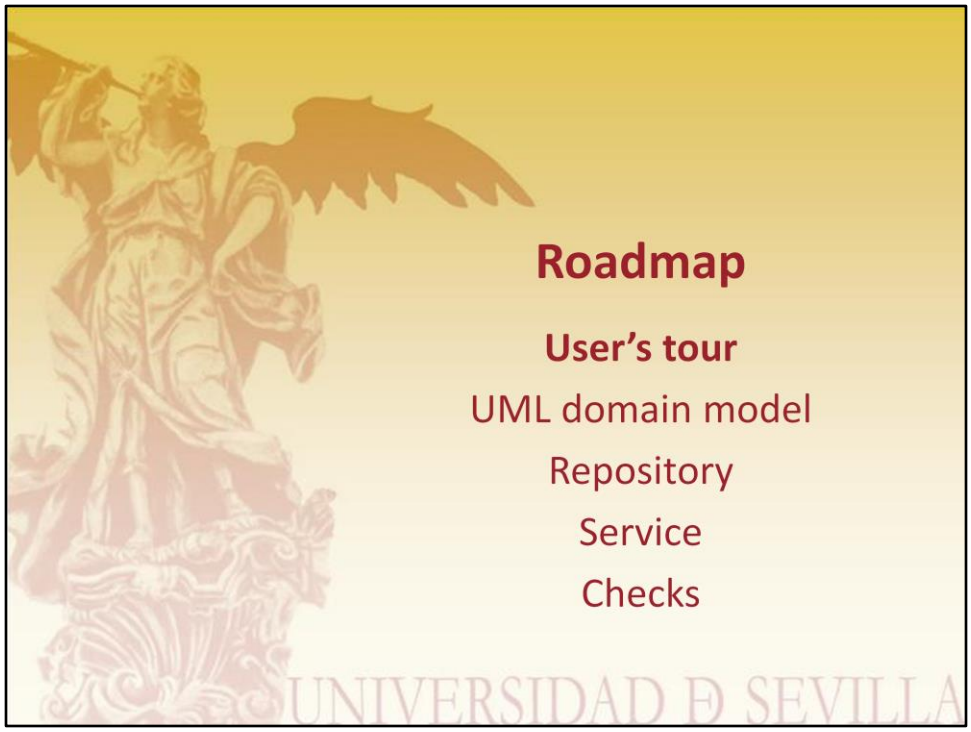
Your first day as a software engineer



Please, realise that this problem is very similar to the problems that you'll have to solve on your first day as a software engineer: your boss will task you with a very simple task like creating a repository or a service building on a specification that will be very similar to ours.

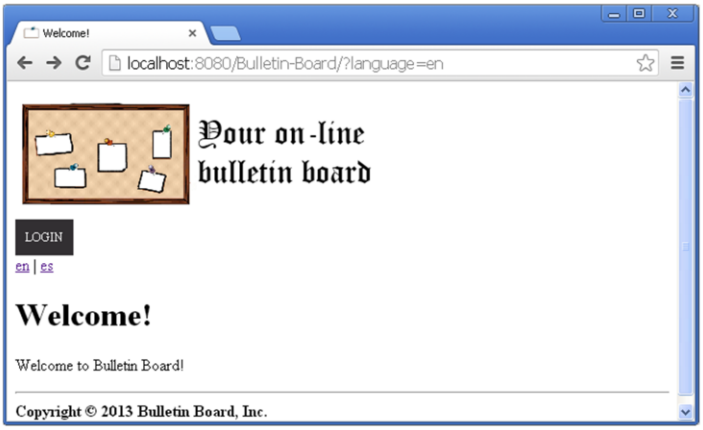


This is our roadmap. It won't be too long since the goal's that you can start working on your project as soon as possible. We'll start with a user's tour; then we'll present the UML domain model, the repository and the service that you have to implement; finally, we'll talk a little about checking them.



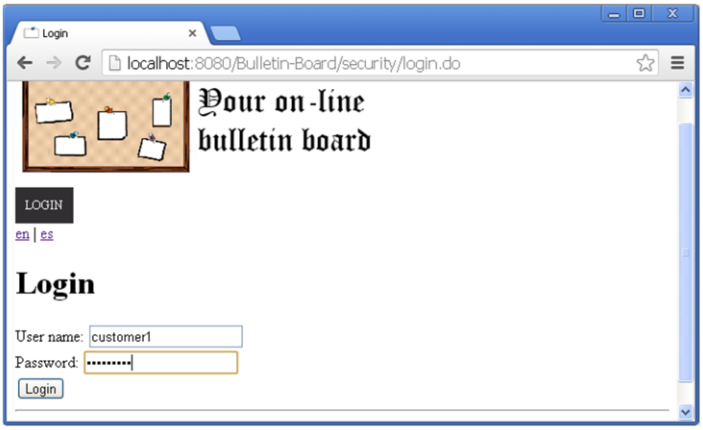
Let's start with the user's tour. In this section, our goal's to present a number of screenshots so that you can have a better understanding of what we expect from this project.

The welcome screen



This is the welcome screen. By now, you should be very familiar with it. It has a logo, a simple menu that just displays “login”, a language bar, a welcome message, and a copyright message.

Logging in as a customer



UNIVERSIDAD D SEVILLA

9

You may log in as a customer using two pre-defined user accounts: “customer1/customer1” and “customer2/customer2”.

The main menu

Once you're logged in, the customer menu will show an option called "Bulletins".

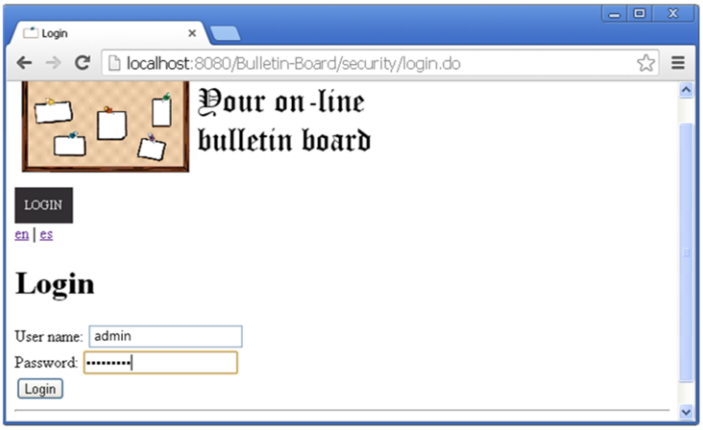
Listing bulletins

This is what you should get whenever you click on “Customer > Bulletins”. It’s a simple list of sample bulletins.

Editing a bulletin

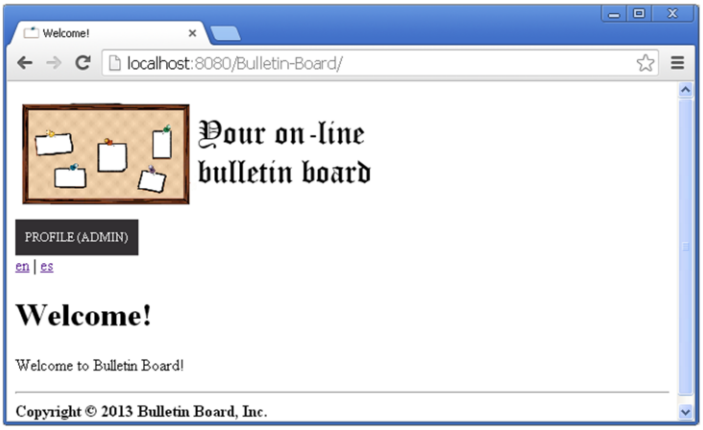
Click on the “Edit” link at the bottom of the first bulletin, for instance. It should result in this screen, in which you can change its attributes, except for the moment, which is set at the server.

Logging in as an administrator

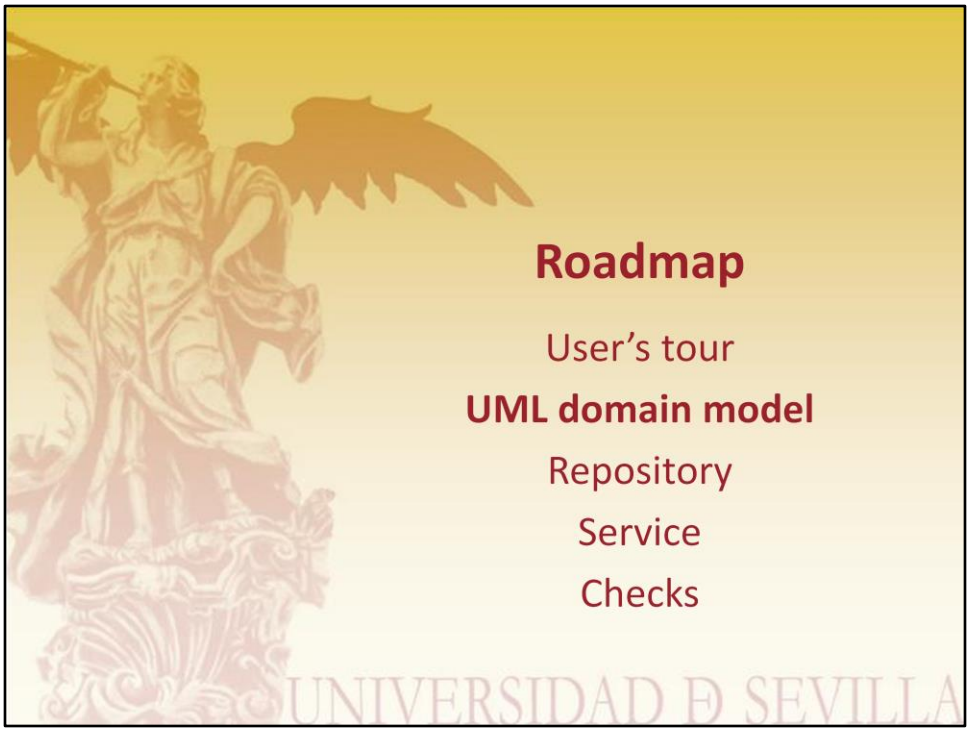


You can also log in as an administrator using the “admin”/”admin” user account.

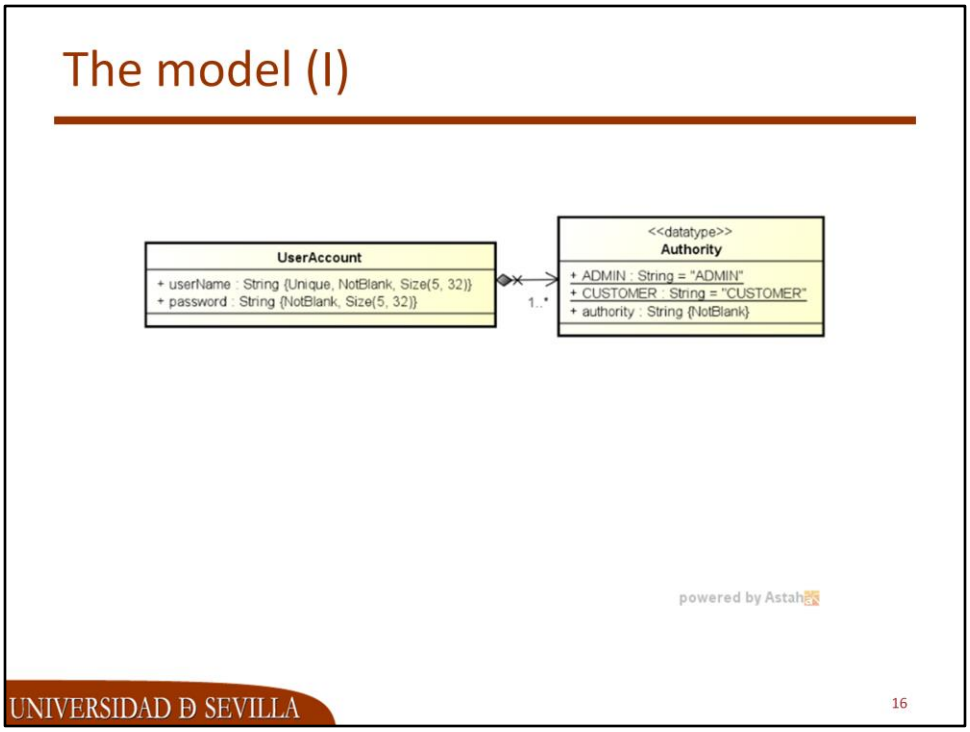
The main menu



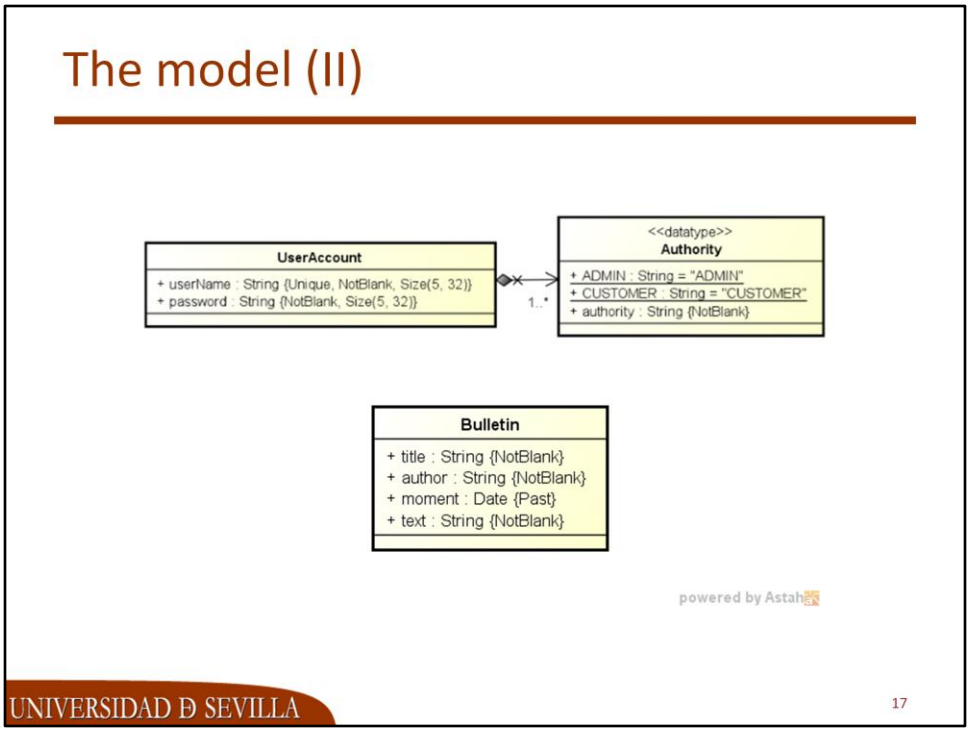
In this case, the main menu doesn't offer any options, except for logout. In other words, we won't implement any administrator's functionalities.



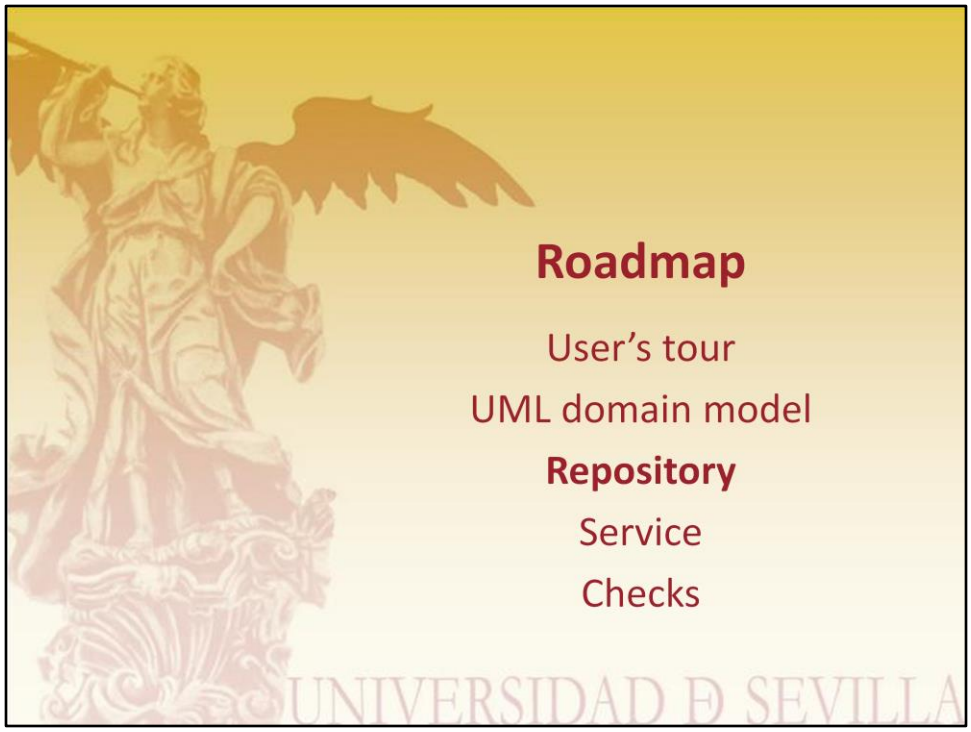
Let's now present the UML domain model.



We'll introduce it in a couple of slides. Classes "UserAccount" and "Authority" should now be very familiar to you. We provided them with the project template. They represent user accounts, which store a username and a hash of a password (please, recall that serious applications never store passwords, but hashes), and have a number of authorities, which is the term that Spring uses to refer to roles. In our project, it's enough to have a couple of authorities: "ADMIN" and "CUSTOMER".



And this is the domain class that models bulletins. It's very simple: every bulletin has a title, an author, a creation moment, and a piece of text that must satisfy the constraints in this slide. Note that class Bulletin isn't related at all with the other classes. This isn't usual at all; typically, we'll have a hierarchy of actors. But this is quite a simple project in which we don't have any actors.



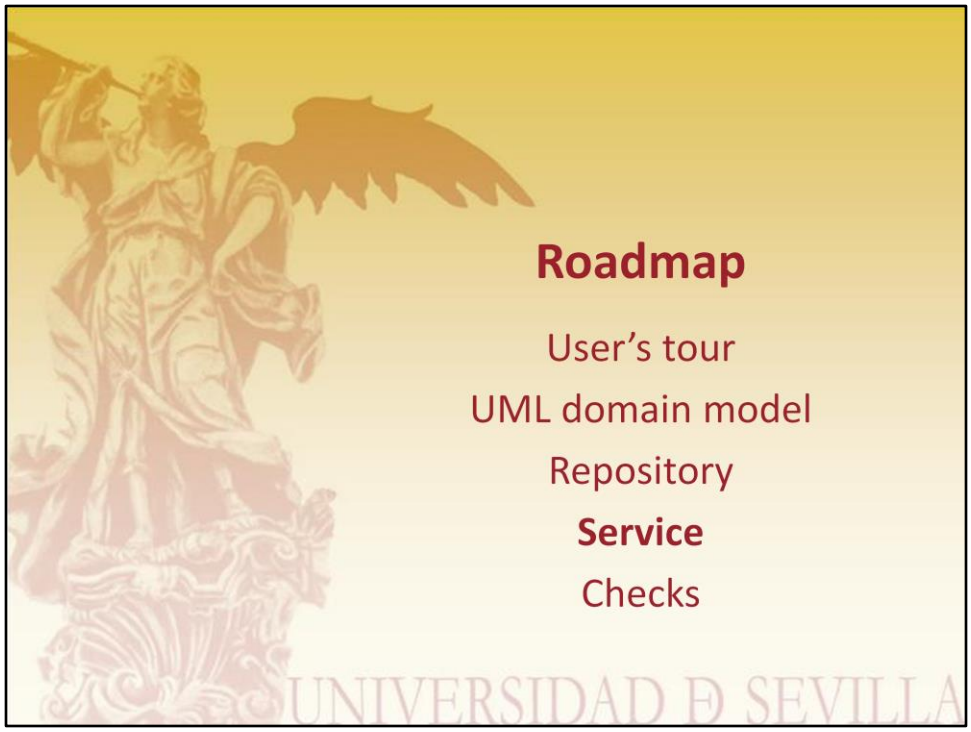
The UML domain model was simple, wasn't it? Let's now provide a little more information on the repository that you have to implement.

The repository

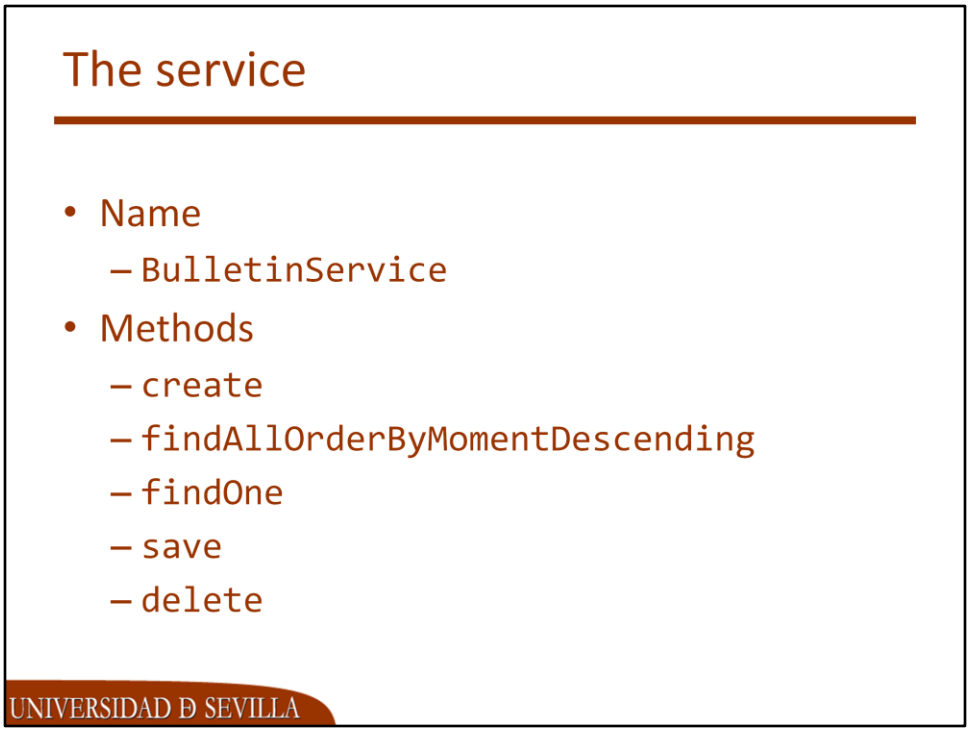
- Name
  - `BulletinRepository`
- Methods
  - `findAllOrderByMomentDescending`

UNIVERSIDAD D SEVILLA

It's name must be "BulletinRepository" and it must provide a single method with the signature in this slide. This method is intended to return a collection with all of the bulletins ordered by the moment when they were created in descending order. It's quite a simple query!

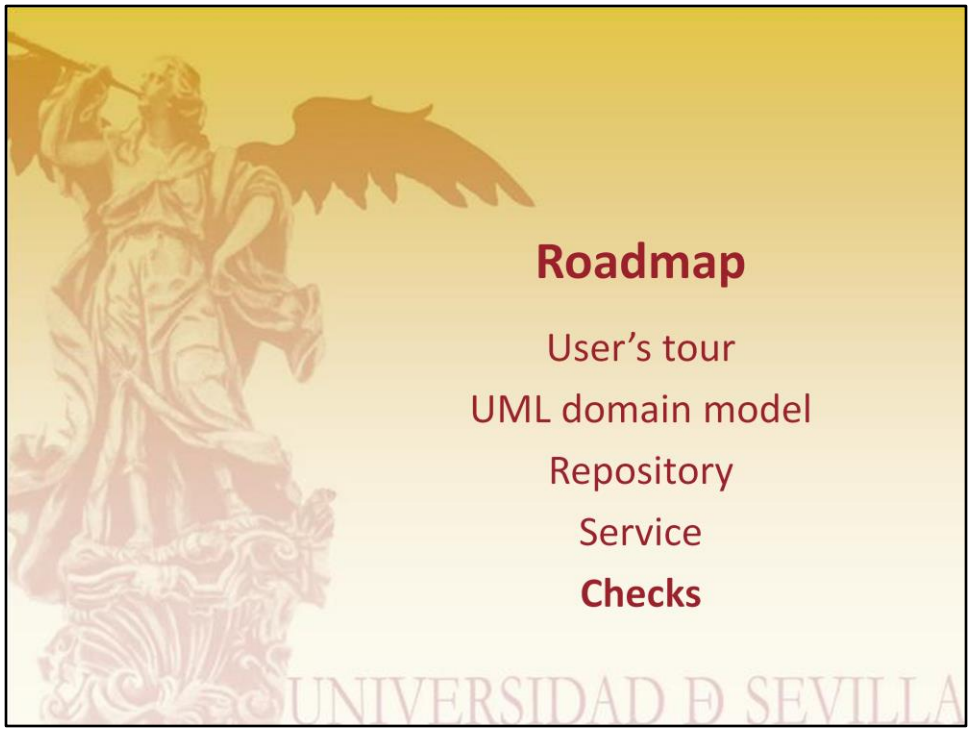


Let's now provide a few details on the service that you have to implement.

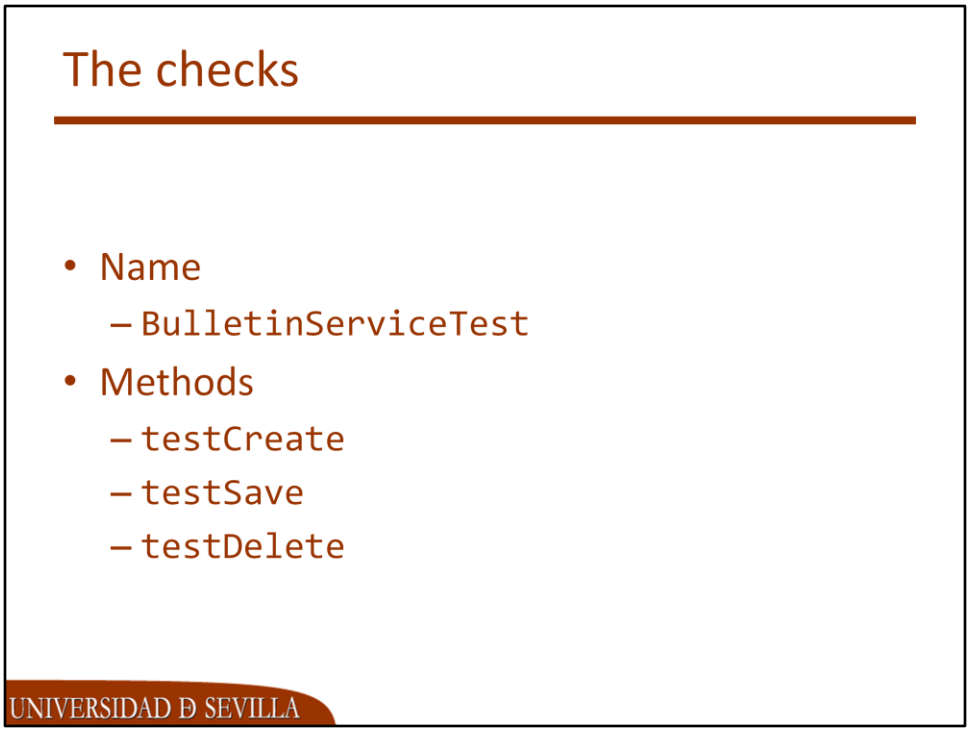


The name of the service must be “BulletinService” and it must implement the following methods:

- “create”: it creates and return a new bulletin.
- “findAllOrderByMomentDescending”: it returns a collection with all of the bulletins ordered by the moment when they were created in descending order.
- “findOne”: it returns the bulletin with identifier `bulletinId’, if any.
- “save”: it saves the bulletin that is passed as a parameter to the database.
- “delete” : it deletes the bulletin that is passed as a parameter from the database.



And finally, let's report a little on the checks that you have to implement.

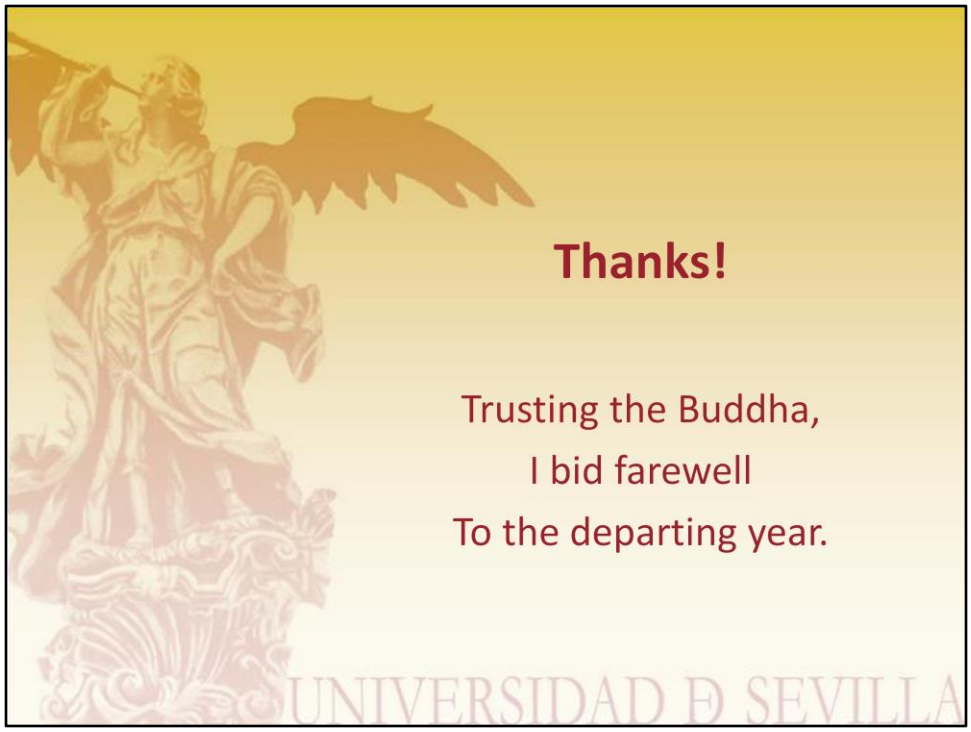


You have to implement a number of checks in a class named “BulletinServiceTest”, namely:

- “testCreate”: check that the “create” method of the service returns an object in which every attribute is null, but the moment.
- “testSave”: check that you can save a newly created bulletin by saving it and then checking that it can be retrieved using method “findAllOrderByMomentDescending”.
- “testDelete”: check that you can delete a newly created bulleting by saving it, then deleting it, and checking that it cannot be retrieved using method “findAllOrderByMomentDescending”.

Ready to work?

Are you ready to work? C'mon!



Thanks for attending this lecture! See you next day!