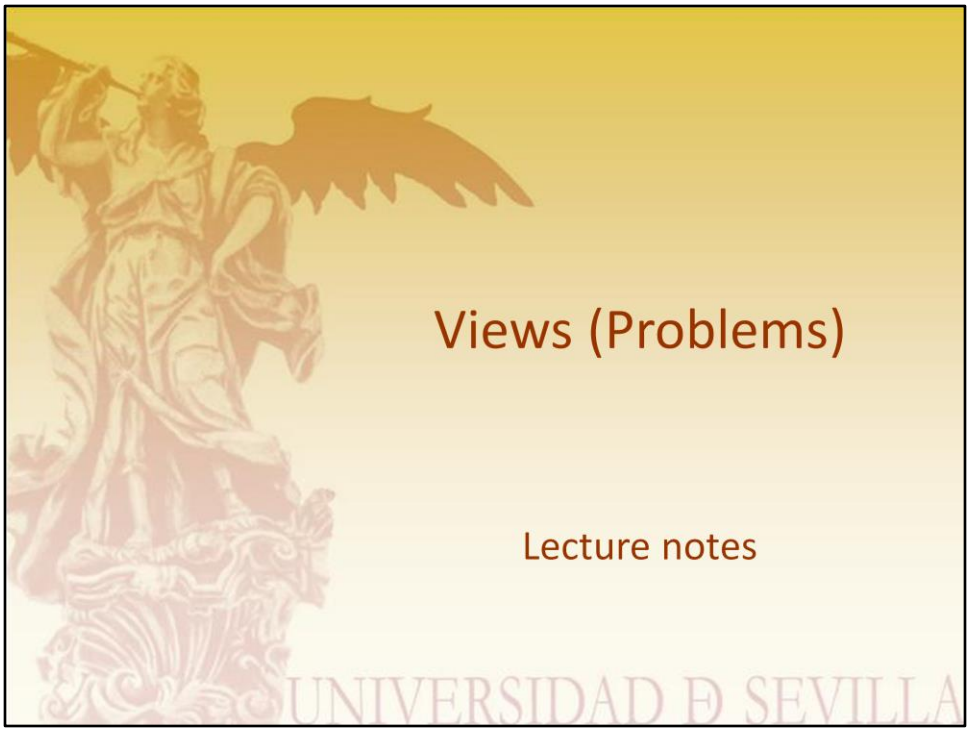


Welcome to this lecture! Today, we'll start putting our knowledge on views into practice.

--

The problem: a greeting-card system

UNIVERSIDAD Ð SEVILLA

2

We'll start with quite a simple project: a greeting-card generator system that people can use to greet their friends and family.

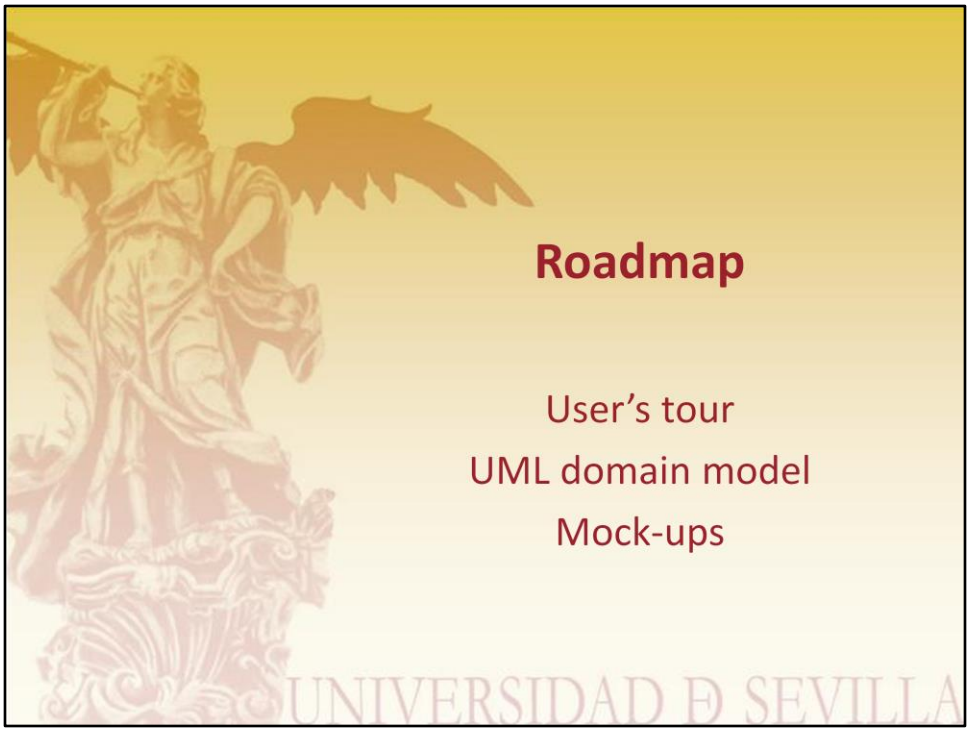
It'll be like riding this bike because we've provided you with a lot of support. The materials that accompany this lecture provide a project called "Greetings" that's almost complete; you only need to provide three views for it to work. Please, note that we won't provide a detailed description of what you have to do to load the project or to set the database up; we assume that you already command these basic steps.

You're constrained, yet!

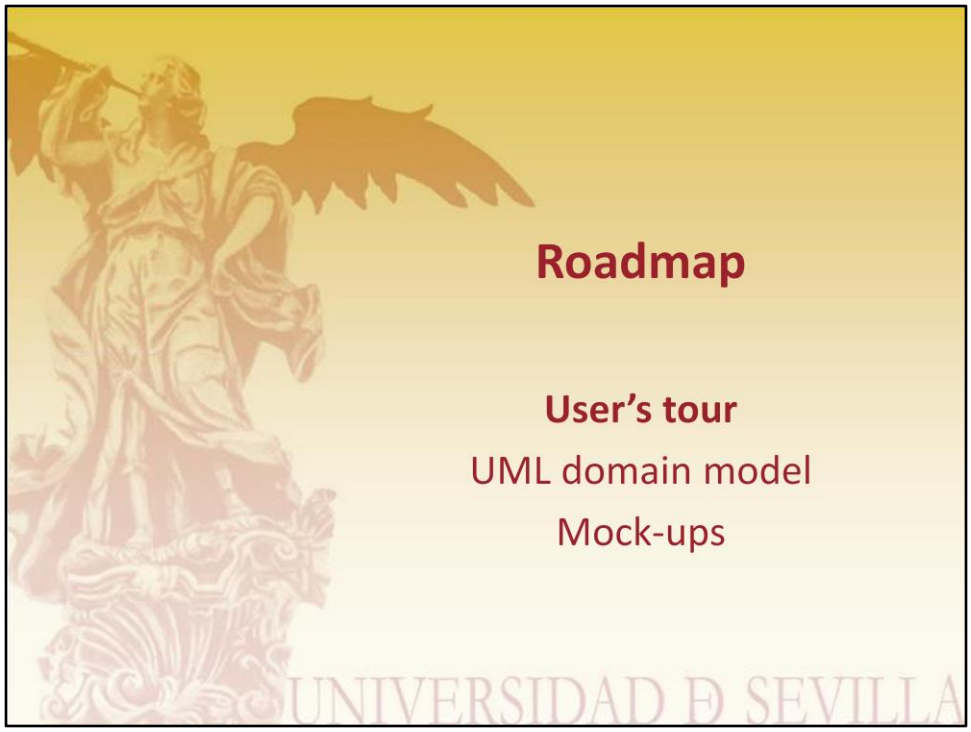
UNIVERSIDAD Ð SEVILLA

Please, note that you're totally constrained. Your only duty in this project is to implement three views; 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!

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 view building on a specification that will be very similar to ours.

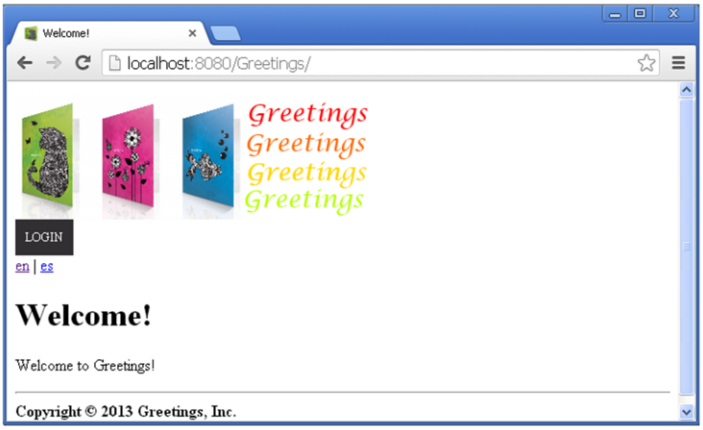


This is our roadmap. It won't be too long since the goal's that you can start working on your project as soon as possible. We'll start with a user's tour; then we'll present the UML domain model; finally, we'll present the mock-ups that you have to implement.



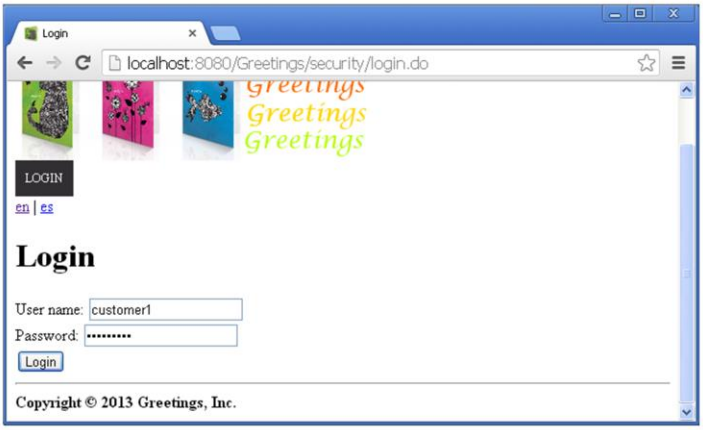
Let's start with the user's tour. In this section, our goal's to present a number of screenshots so that you can have a better understanding of what we expect from this project.

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, and a welcome message.

Logging in as a customer

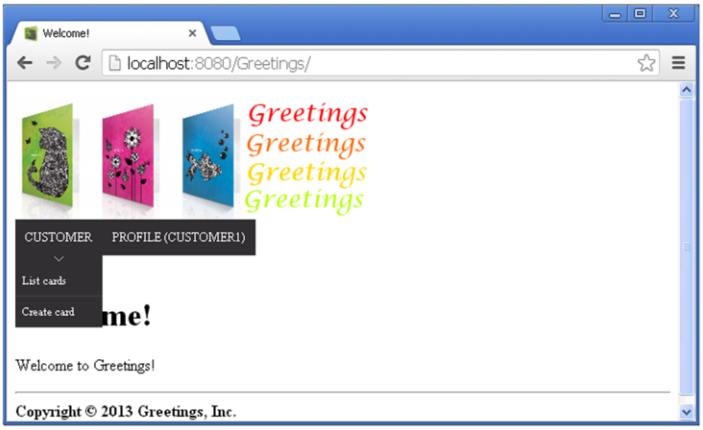


UNIVERSIDAD D SEVILLA

9

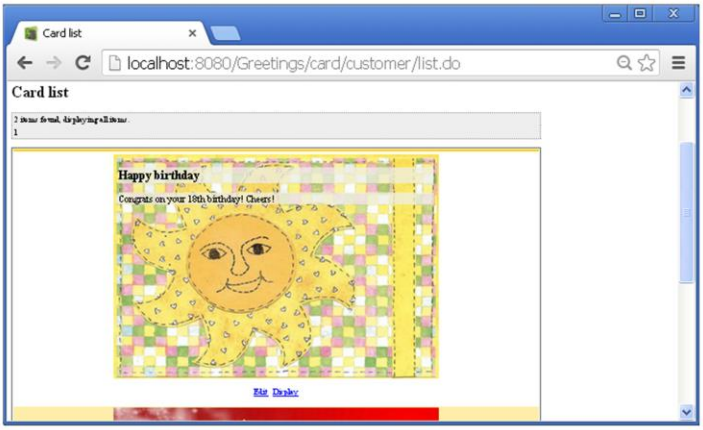
You may log in as a customer using two pre-defined user accounts: “customer1/customer1” and “customer2/customer2”.

The main menu



Once you're logged in as a customer, the menu will show an option called "List cards" and another one called "Create card".

Listing cards



UNIVERSIDAD D SEVILLA

11

This is what you should get whenever you click on “Customer > List cards”. It’s a simple list of sample greeting cards. Note that every card has a title, a piece of text, and a background; in addition, there are a couple of links to edit or to display them.

Editing a card

Click on the “Edit” link at the bottom of the first card, for instance. It should result in this screen, in which you can change its attributes, namely: the title of the card, its text, and its background, which you can select from a dropdown list.

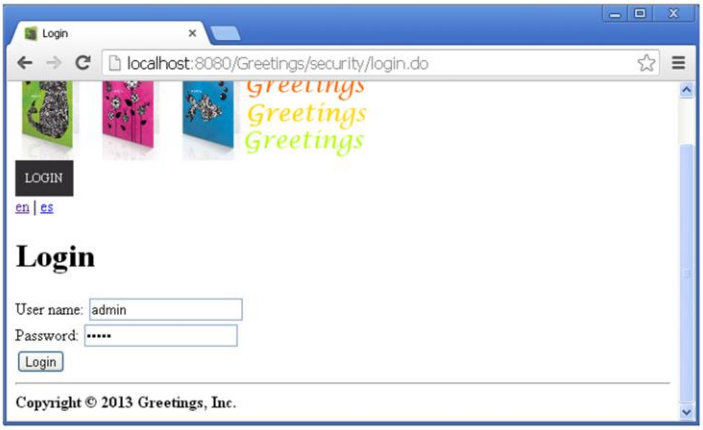
Displaying a card

UNIVERSIDAD D SEVILLA

13

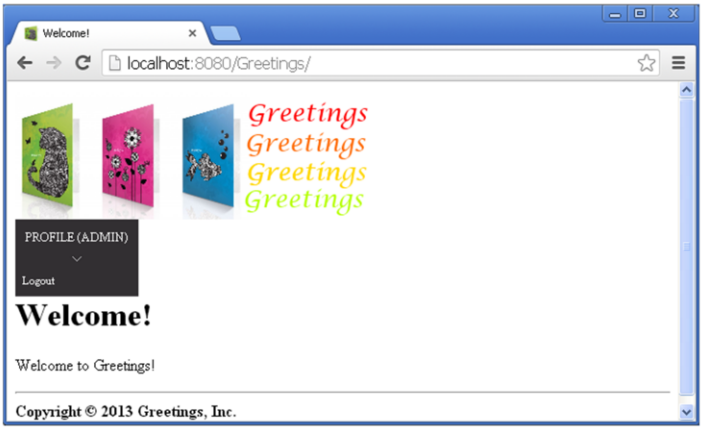
If you click on the “Display” link at the bottom of the first card, you’ll get the card in a new tab that is publicly accessible. That is, you may copy the URL and send it to a friend by email. Check that this works by making another browser to the URL you get. (It’s important that you use another browser so as to avoid interferences; a Chrome incognito tab might work, but two incognito tabs won’t work.)

Logging in as an administrator

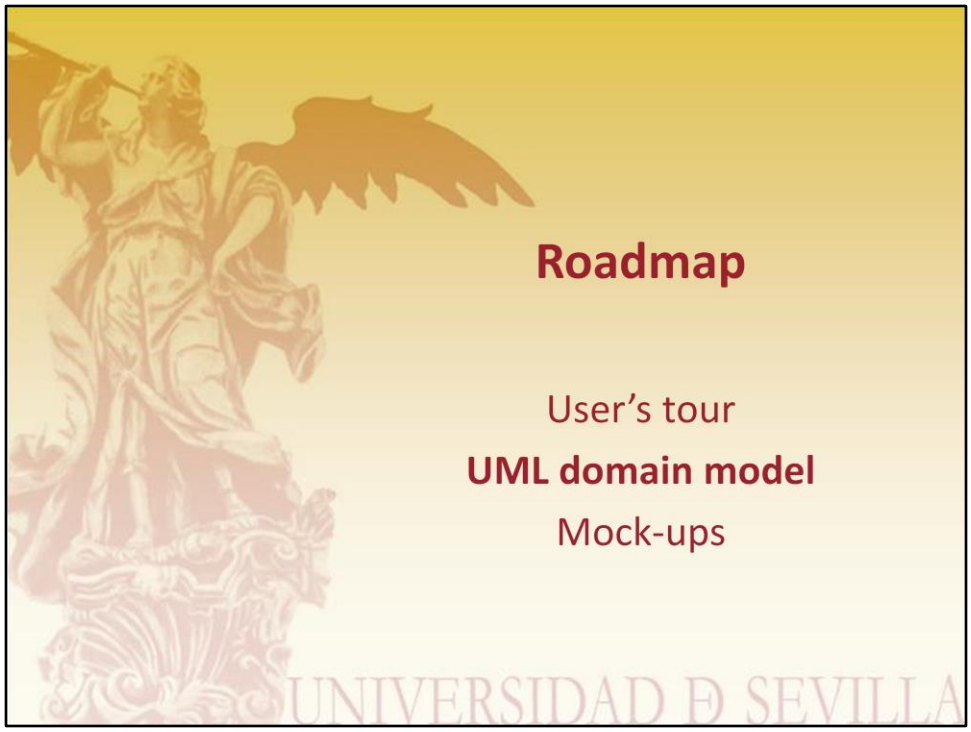


You can also log in as an administrator.

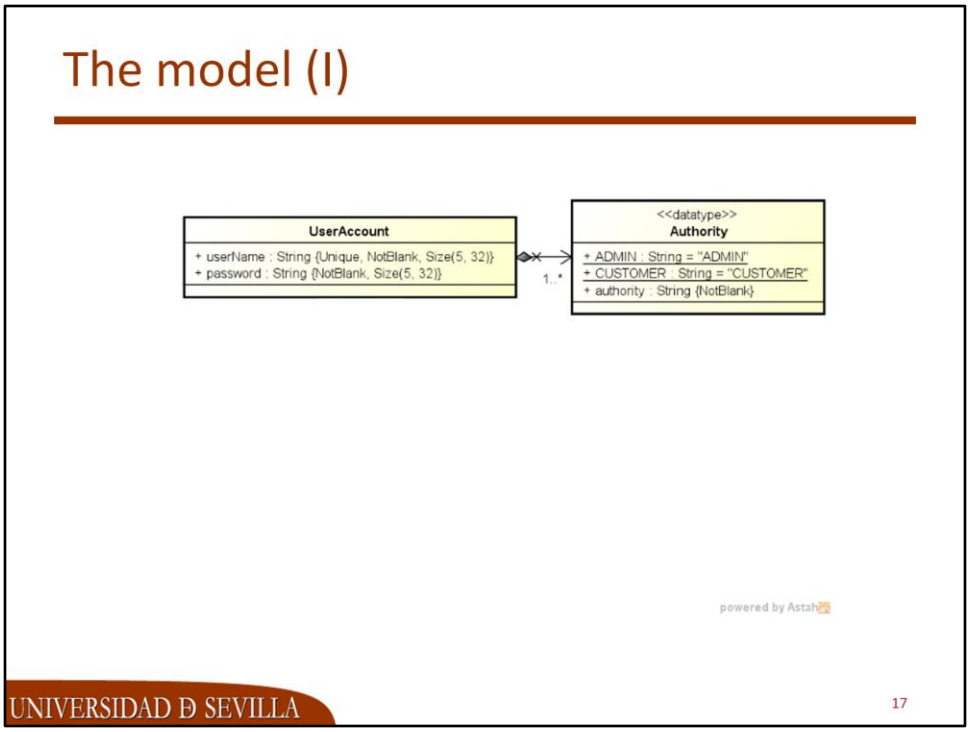
The main menu



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.

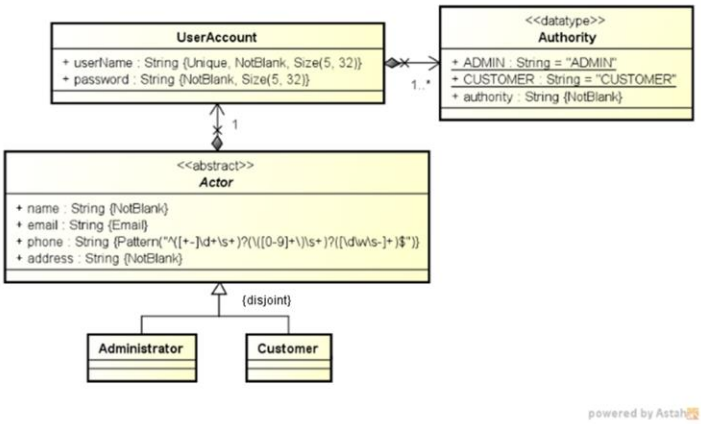


Let's now present the UML domain model.



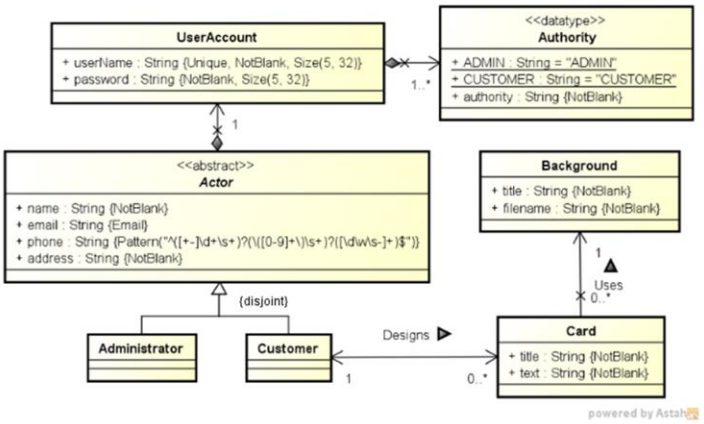
We'll introduce it in a couple of slides. Classes "UserAccount" and "Authority" should be very familiar to you. We provided them with the project template and we've used them a lot of times before. They represent user accounts, which store a username and a hash of a password (please, recall that serious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

The model (II)

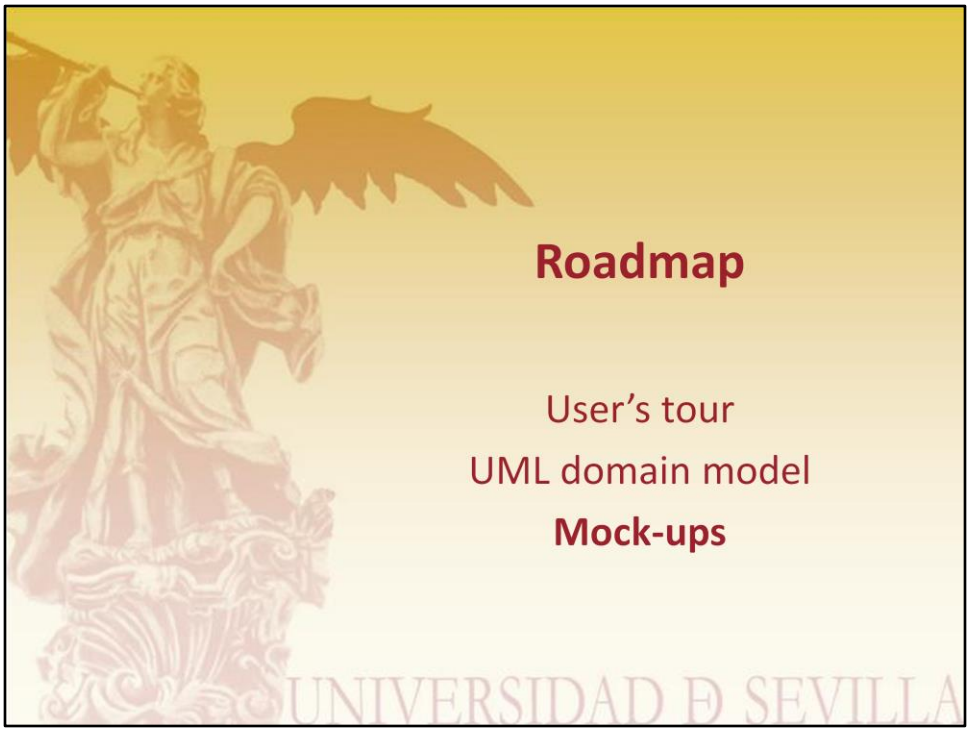


This slide shows the actor model, including a superclass called “Actor” that models the attributes that are common to both administrators and customers.

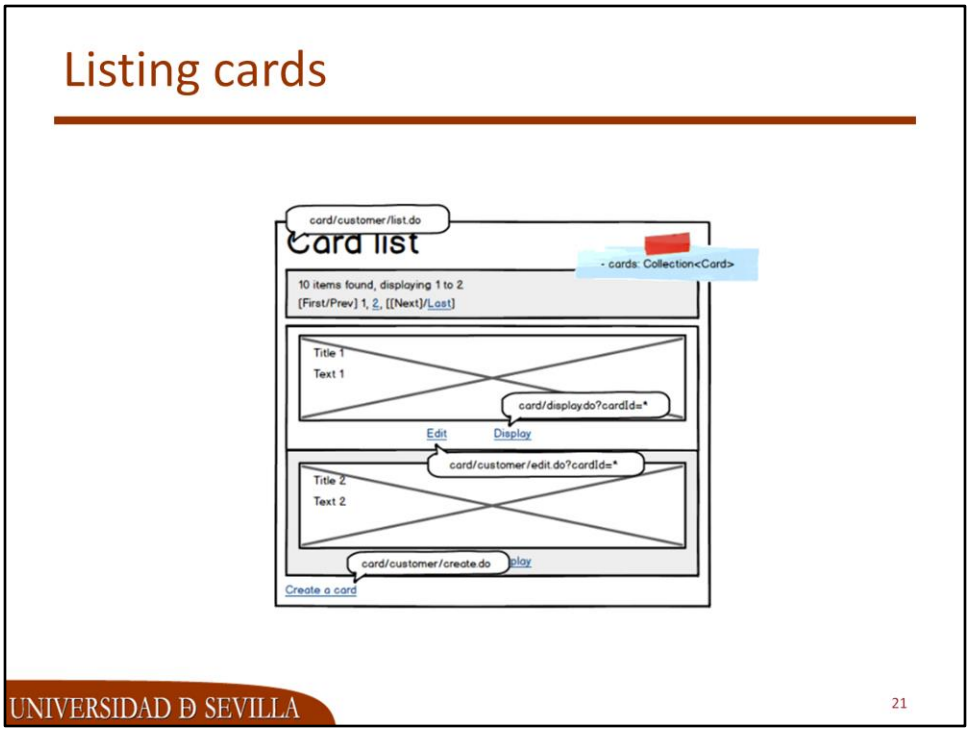
The model (III)



And finally, this is the part of the model that involves the cards. Note that every card has a title attribute and a text attribute, and is involved in two associations, namely: association "Uses" helps model the background used for a given card; association "Designs" establishes a relationship between the customers of the system and the cards that they create. Every background has a title to describe it and a filename that refers to a file in folder "webapp/images/backgrounds". Please, peek at our backgrounds; they are very nice.



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

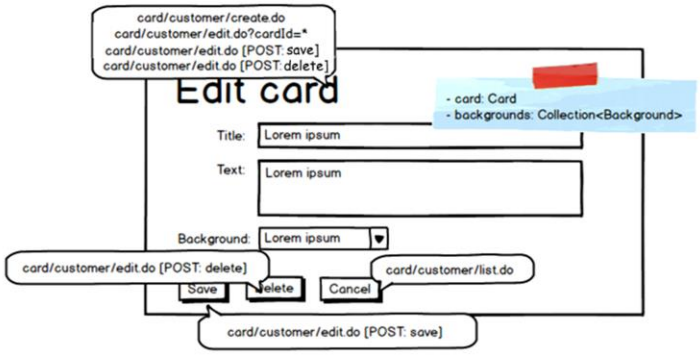


This is the mock-up to list cards. It's a typical listing, but instead of presenting the information in columns, we present it more aesthetically. Note that every row of the listing shows a different card; the image of the card is shown in the background (images are represented using big crossed boxes in the mock-ups), and the title and the text is shown on top of the background. (This is commonly referred to as overlaying a piece of text on an image.) Below every card there's a link to edit and display it; below the list, there's an additional link to create new cards. This view requires the model that is presented in the cyan sticky note, which consists of a single variable called "cards" that provides the collection of cards to be displayed in the listing.

NOTE: don't worry if you don't know how to overlay a piece of text on top of an image; this is the HTML code you have to use:

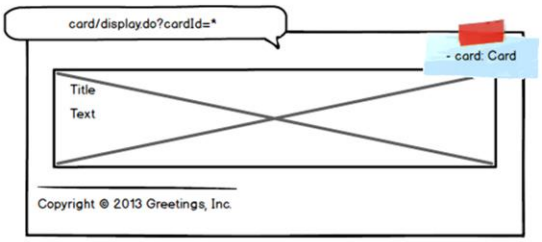
```
<div style="position: relative; width: 649px; height: 448px; margin-left:auto; margin-right:auto;">
  <img src="⟨link-to-the-background-image⟩" />
  <div style="position: absolute; top: 10px; left: 10px; right: 10px; bottom: 10px;">
    <h1 style="font-size: 25px; background-color: rgba(240, 240, 230, 0.75);">
      ⟨title-of-the-card⟩
    </h1>
    <p style="font-size: 18px; background-color: rgba(240, 240, 230, 0.75)">
      ⟨text-of-the-card⟩
    </p>
  </div>
</div>
```

Editing cards



This is the mock-up to edit cards. Note that it's a typical edition form in which the user must provide a title, a text, and select a background in a dropdown list. The model provides the card to be edited in variable "card" and the list of backgrounds available in variable "backgrounds".

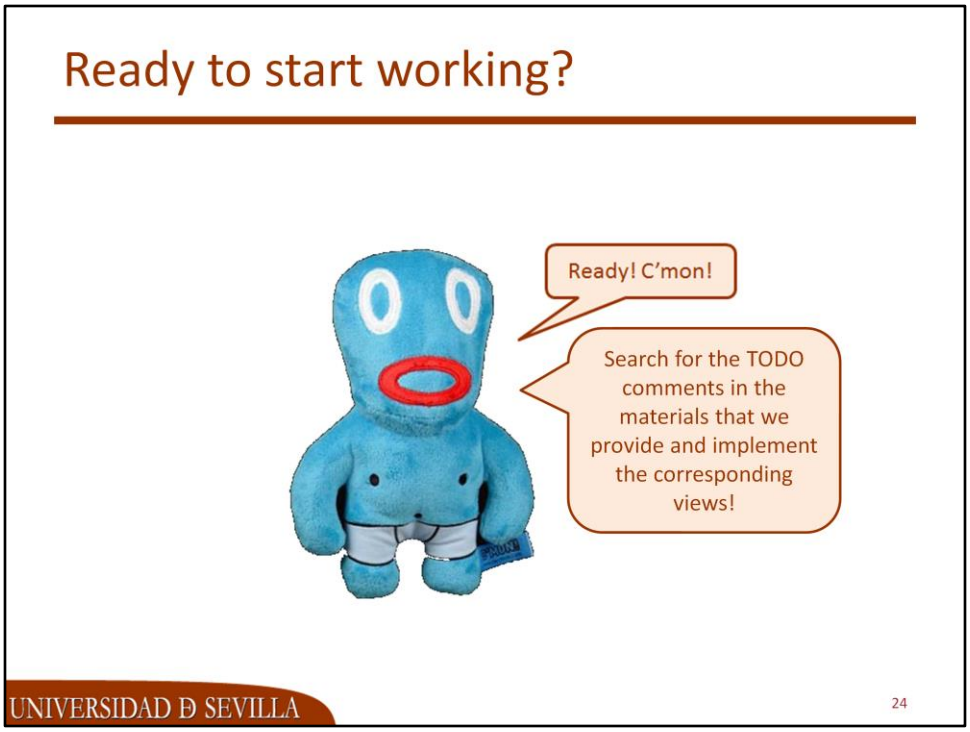
Displaying cards



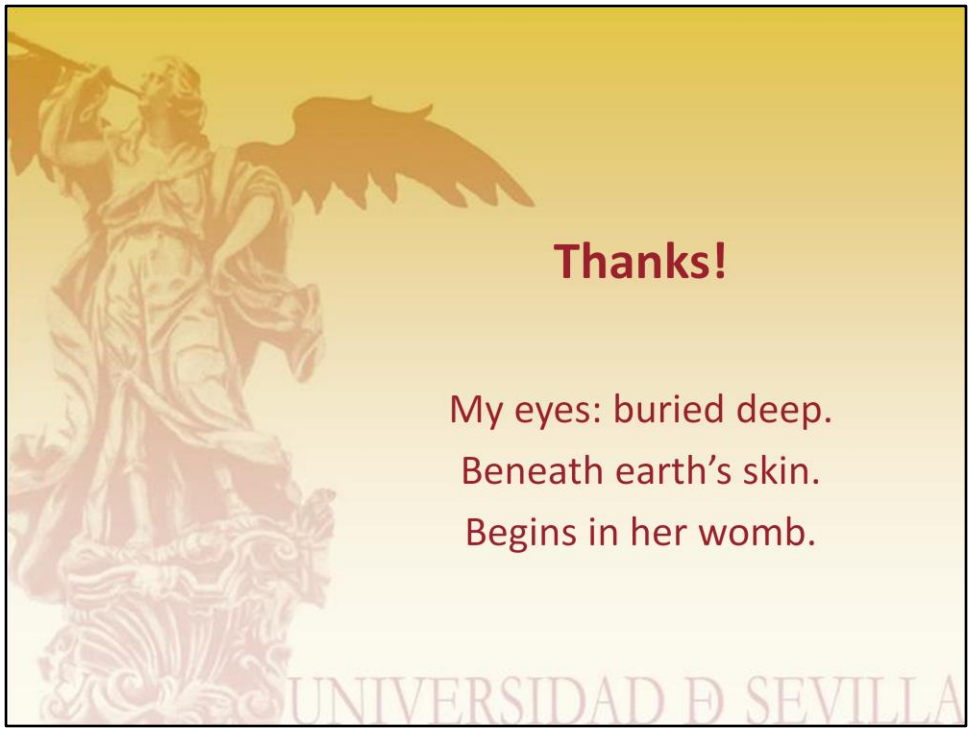
UNIVERSIDAD D SEVILLA

23

And this is the third view that you have to implement: it's a very simple view to display a card. You may share the link to this view since it's publicly available.



Ready to start working? C'mon!



Thanks for attending this lecture! See you next day!