Technical University of Moldova
Inginerical department S.A.

Report

№: 8

AMOO

Subject: Component Diagrams. Project setup. Top-down and bottom-up design.

Author: Terman Emil FAF161
Prof: M. Gavrilita

Chisinau 2018

Theory

Component diagrams

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose? Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Benefits

– imagine the system's physical structure;
– emphasize the service behavior as it relates to the interface;
– pay attention to the system's components and how they relate;

Top-down and bottom-up design

Top-down and **bottom-up** are both strategies of information processing and knowledge ordering, used in a variety of fields including software, humanistic and scientific theories (see systemics), and management and organization. In practice, they can be seen as a style of thinking, teaching, or leadership.

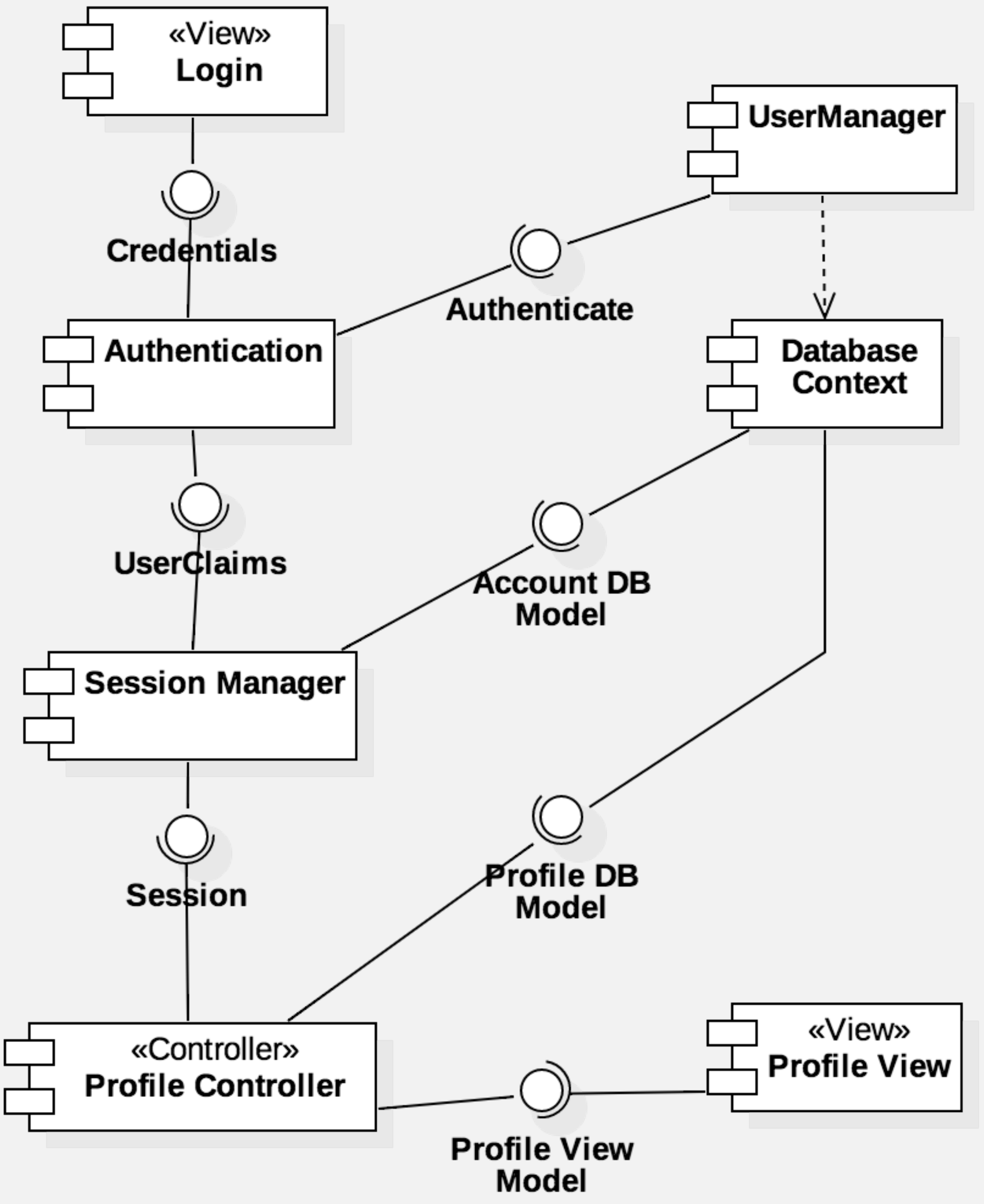
– a **top-down** approach (also known as stepwise design and in some cases used as a synonym of decomposition) is essentially the breaking down of a system to gain insight into its compositional sub-systems in a reverse engineering fashion. In a top-down approach an overview of the system is formulated, specifying, but not detailing, any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. A top-down model is often specified with the assistance of "black boxes", which makes it easier to manipulate. However, black boxes may fail to clarify elementary mechanisms or be detailed enough to realistically validate the model. Top down approach starts with the big picture. It breaks down from there into smaller segments;
– a **bottom-up** approach is the piecing together of systems to give rise to more complex systems, thus making the original systems sub-systems of the emergent system. Bottom-up processing is a type of information processing based on incoming data from the environment to form a perception. From a cognitive psychology perspective, information enters the eyes in one direction (sensory input, or the "bottom"), and is then turned into an image by the brain that can be interpreted and recognized as a perception (output that is "built up" from processing to final cognition). In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, by which the beginnings are small but eventually grow in complexity and completeness. However, "organic strategies" may result in a tangle of elements and subsystems, developed in isolation and subject to local optimization as opposed to meeting a global purpose;

Component Diagrams

In *Figure 1* is represented the component diagram of Login. I tried to keep to the MVC patern. With that being said, the user interacts with the **Login** *View*, which sends the *Credentials* to the *Authentication*. It then communicates with the *UserManager* to authenticate and then form the user's *Claims*. A *Session* is formed and it is used for further user identification. Finally, the *Profile View* is fed with a *Profile View Model*.

Figure 2 represents the components of commenting. It interacts with the *Authorization Manager* to check if the user is able for such an action. To check if the content does not contain anything inappropriate, the contents are sent to a checker. If it is found to have problems, an email notification can be sent to an admin trough the *Email Service* and a *Report* can be registered in the Database.

In *Figure 3* is shown the components needed for a Photo Uploader. First, it gets the Logged User to link the user ID to the photo. Then, it interacts with the *Tag Manager* to find possible tags on the uploaded image. It must also check for Inappropriate content which is done in the *Inappropriate Content Manage*. Again, if a problem is found, an email notification can be sent.



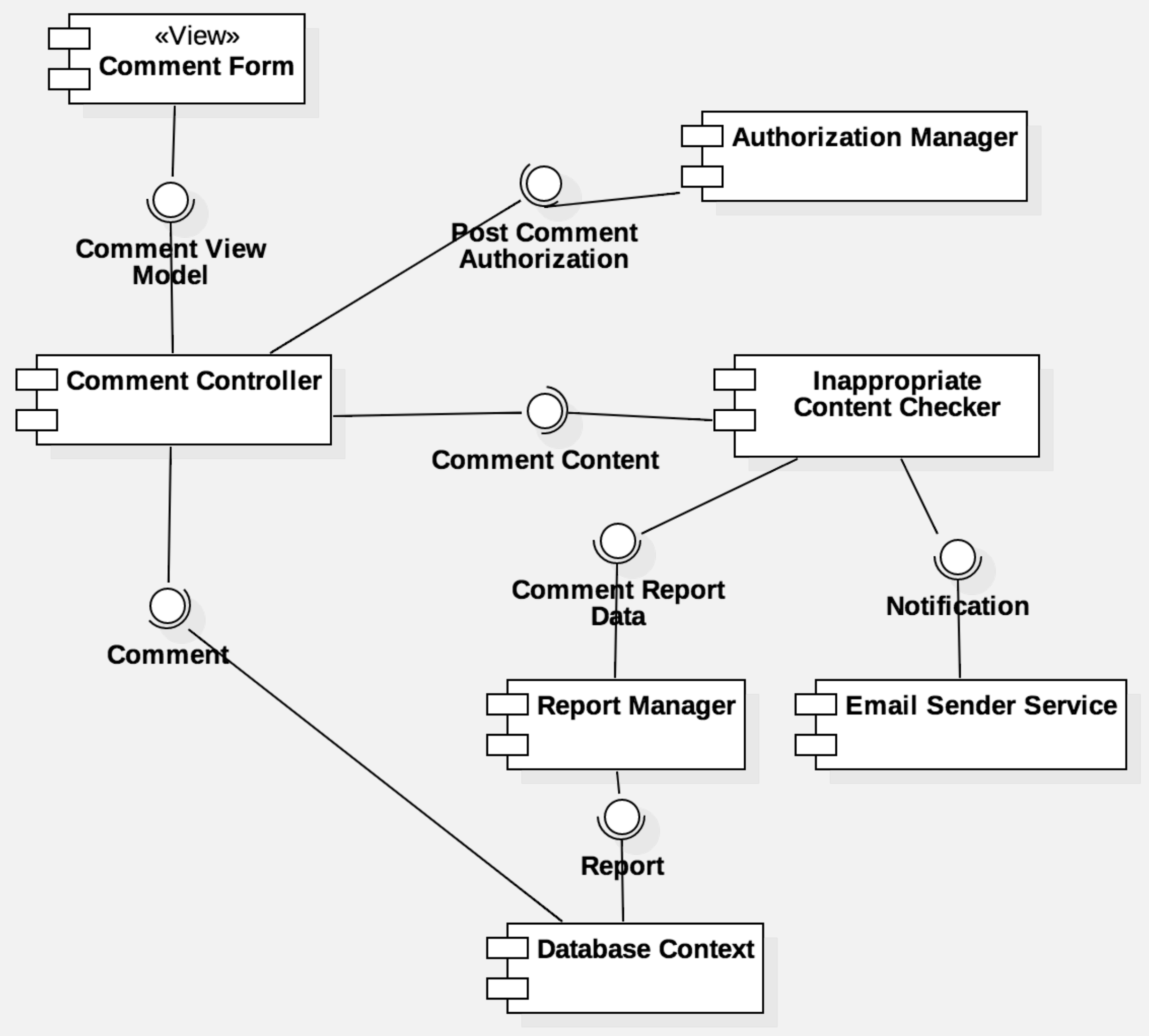
Img 1: Login Component Diagra

Project Setup

To work on this project, I would require Visual Studio, because I would write it in ASP.NET Core 2. I would definetly need git to work with my team. As for the database, I would install SQLite.

For testing, I would need multiple browsers, like Chrome, Firefox, Opera, Sefari, Explorer. For testing on mobile platforms, I would need multiple types of phones: Android, IOS, etc.

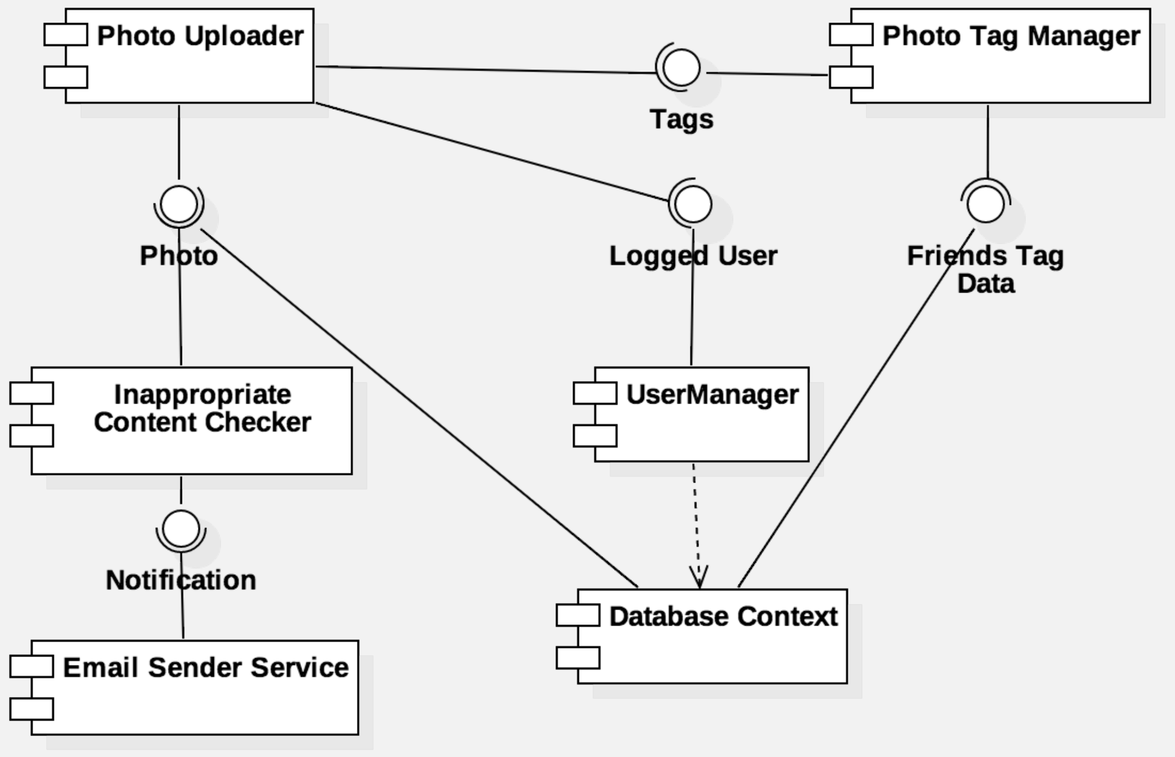
To check if the POST/GET and other requests work, I would use *Insomnia REST*.



Img 2: Login Component Diagra

Conclusion

Component diagrams, are very helpful to get a general idea about what parts of the system interact with each other, and what components can be reused. This type of diagrams don't enter too deep into details. Rather than *Class Diagrams*, it offers a much more general view of the project.



Img 3: Upload Photo Component Diagra