Linnéuniversitetet
Kalmar Växjö

Assignment 4

Architecture Design

Författare: Jonathan Alklid, Helena Tevar

Examinator: Jesper Andersson
Termin: VT 20
Ämne: Software Architectures
Kurskod: 2DV604

Linnéuniversitetet
Kalmar Växjö

View: System Security

Overview

This view will describe the strategies followed to resist attacks and cover security concerns in our system in relation with identification, authentication and authorization of the users. The pattern used in this view is a Role Rights Definition pattern[1] that will be implemented in the context of a Broker communication pattern[1]. We will associate users with roles and roles with permissions that will allow the system to hold a large number of users with shared resources that will be accessed by different privileges. This pattern will improve the maintenance of the system users, since there are less roles than users. Roles should be easy to change and for that reason, easy to maintain.

Tactics Used

Identify actors: The system should be able to identify where the source of the request is coming from. For this is recommended the use of technologies that allow our system to check the inputs it receives: Cookies, CSRF: Cross site request forgery.
Authenticate users: The system should provide functionality that ensures that the users are who they are by User/Password identification or 2 step authentication.
Authorize users: Provide functionality that ensures that a user can only access the functionalities related to their role. A user may own different roles, and a role may have multiple functionalities. The system should provide a role based mechanism.
Encrypt communication: The flow of data between user and server should be protected by any means of encryption. The system should use TLS/SSL protocols to provide communication security. The system should add encryption to password, salt and hash, and by any means save non encrypted passwords in the system persistence.
Lock computer: The system should lock a user account after a certain amount of repeatedly failed login attempts, or repeated attempts at accessing a restricted resource (brute force protection).
Limit access: The system should limit access through the use of a proxy and a firewall that serve to block ports, restrict protocols, and reject access to unknown and/or suspicious connections.
Limit exposure: The system should limit its external exposure through the use of a firewall and a broker-component with a restrictive external API interface.

Audience

Stakeholder	
Architect	Responsible of the development of the architecture

[1] See View: Flexible Communication

Project Manager	Responsible for the planning and allocating
Designer	Responsible of designing and meet architectural requirements
Implementer	Responsible for the development of elements
Maintainer	Responsible for fixing bugs and providing enhancements
Tester	Responsible for test and verification

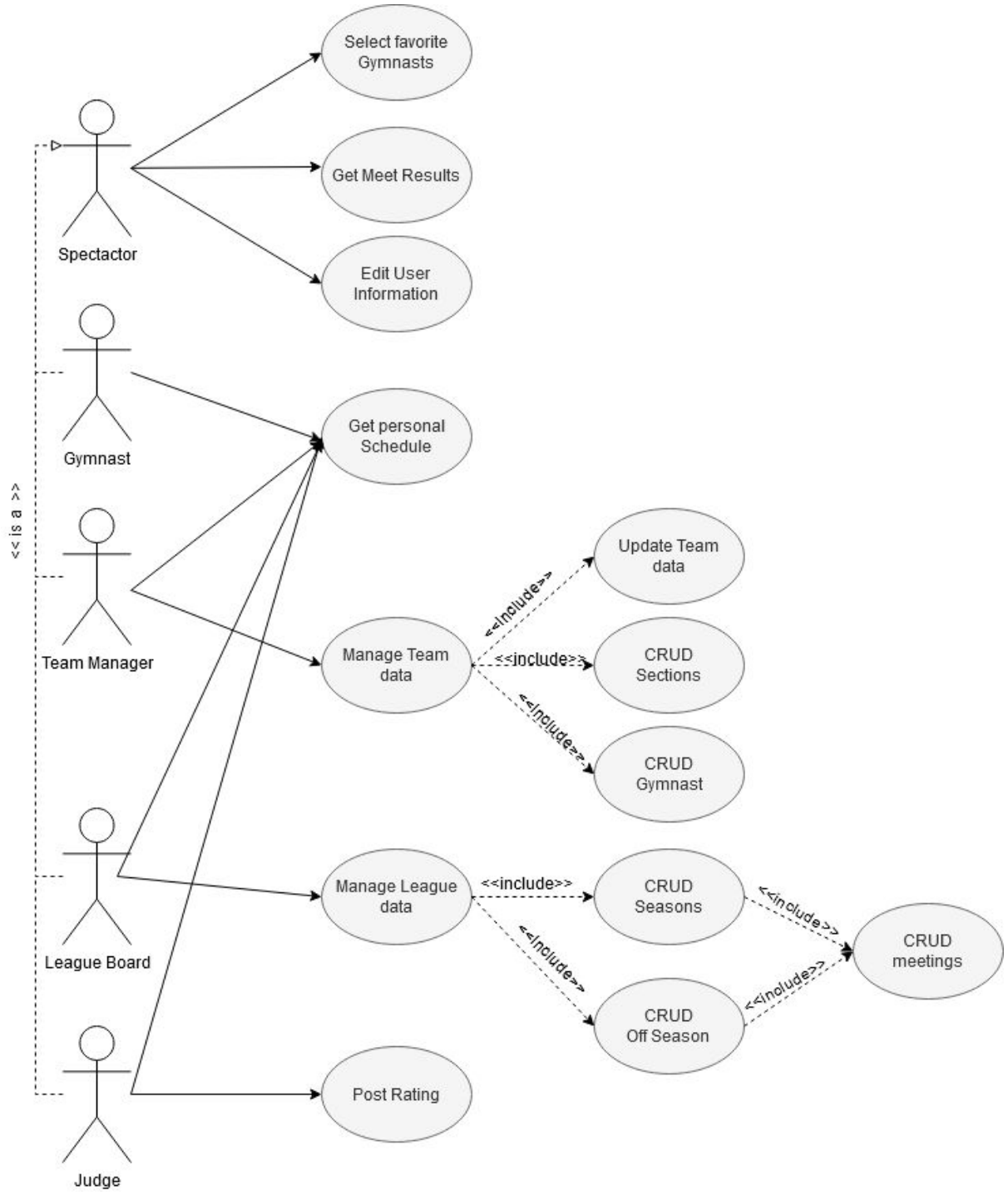
Models

This section contains the models provided below:

Model
Use Case diagram
Sequence diagram
Deployment diagram

Use Case diagram

The model below describes which user roles have privileges to access certain functionalities. It has to be followed when describing and implementing roles for the system.



The different roles are defined as:

- Spectator: User that is registered in the system with the lowest access role. All other roles include the functionality of spectator roles.
    - is allowed to modify their personal information

    - is allowed to get the results of the meets
    - is allowed to select favourite gymnasts
- Gymnast: User that is a member participant of a team.
    - is allowed to get their personal schedule
- Team Manager: User with responsibility over teams.
    - is allowed to update information on their team
    - is allowed to create, modify or delete sections in their team
    - is allowed to add, modify or delete gymnasts in their team
    - is allowed to get the schedule for their gymnasts
- League Board: User with responsibility over the League.
    - is allowed to create, modify or delete new seasons
    - is allowed to create, modify or delete new off seasons
    - is allowed to create, modify or delete meets for seasons and off seasons
    - is allowed to get the schedule for the teams
- Judge: User with responsibility over the rating of meets and gymnasts.
    - is allowed to rate a gymnast meet.
    - is allowed to get the schedule of their meets.

Element Catalogue

Elements

- User: Stakeholder that has access to our system
- Role: Status that expects certain behaviour from the system
- Privilege: Behaviour or functionalities only expected for certain roles
- Permission: Module that performs checks on roles and provide allowance to privileges

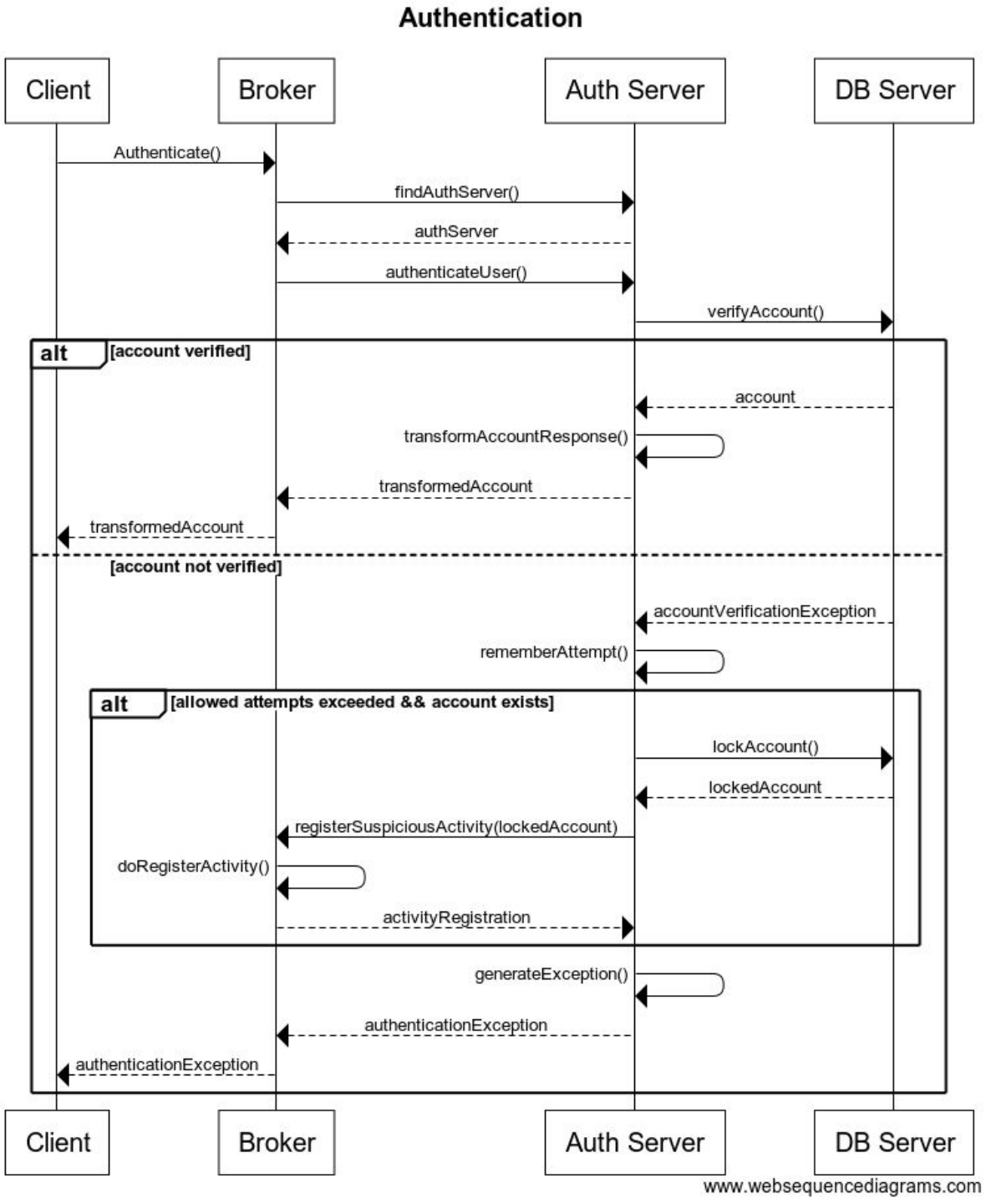
Relations

Roles and privileges allow access to the users

Sequence diagrams

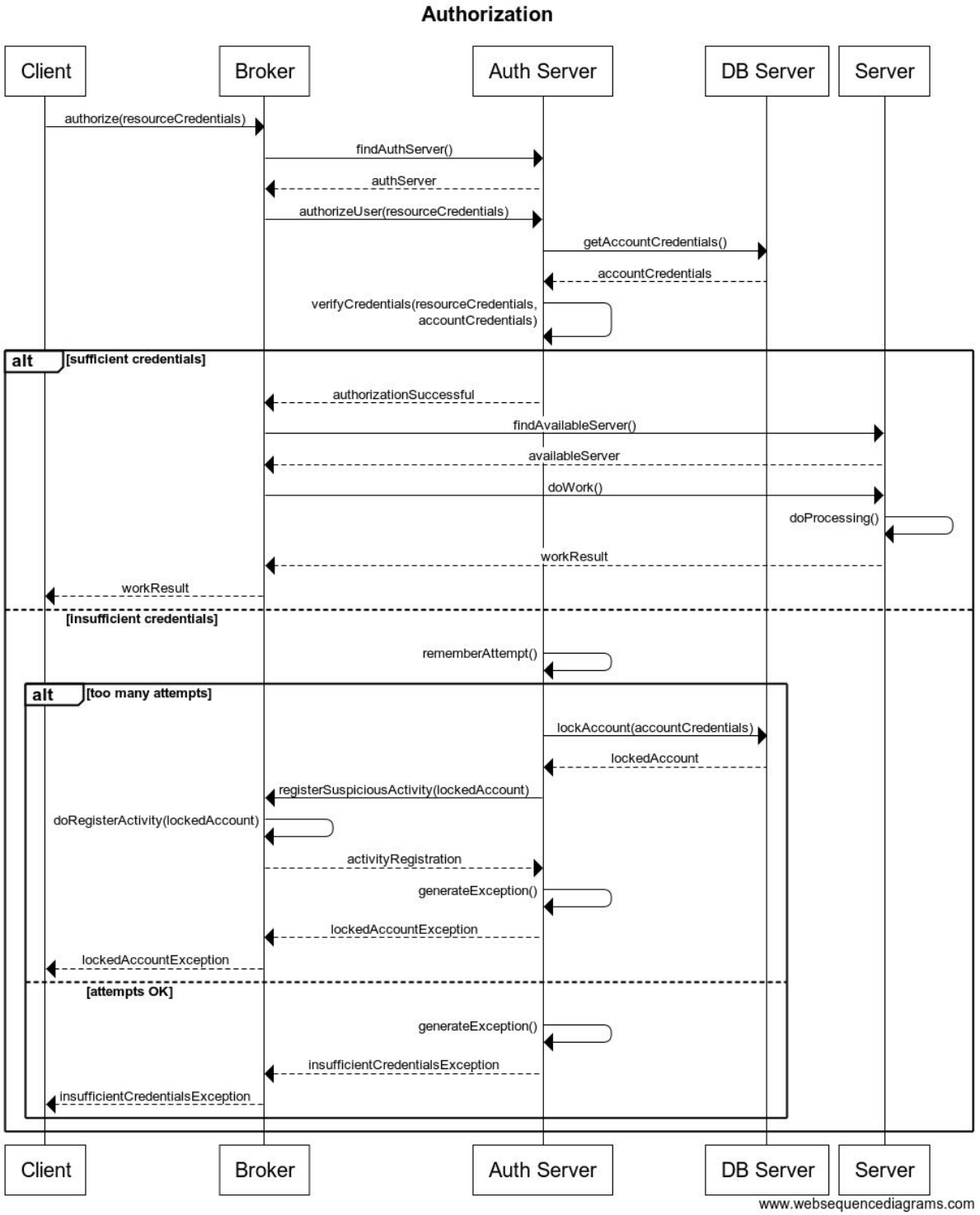
These models show the communication between components that are relevant for this view. The first one shows the communication on authentication, the last one shows the communication on authorization.

Authentication



Authentication starts when a client wishes to authenticate itself. The broker finds an available authentication/authorization server and then attempts authentication. The authentication server connects to the database and checks if the user exists and the details are correct. If they are, the authentication server will return a transformed account that the client can store for future requests. The transformed account removes sensitive information and only exposes safe details the clients can keep for session maintenance. If they are not, the authentication server returns an authentication exception that prevents the user from logging in. Additionally, if the user fails to authenticate several times in succession, the account will be locked and a system administrator will be notified. The notification process has been omitted to conserve space.

Authorization



Authorization begins when a user attempts to access a restricted resource. The broker finds an available authentication/authorization server and then attempts authorization. The auth server connects to the database and retrieves the user's credentials after which they are compared against the credentials required for the requested resource. If the credentials are sufficient, the broker is notified after which the broker finds an available server for work, waits for it to finish the requested work, and then returns the result to the client. If the credentials are not sufficient, the auth server will remember the attempt and if the attempts exceed the allowed number, the account will be locked, the administrators will be notified,

and an exception will be returned informing the user that their account has been locked. If the attempts have not been exceeded, an exception notifying the user of their insufficient credentials will be generated and returned.

Element Catalogue

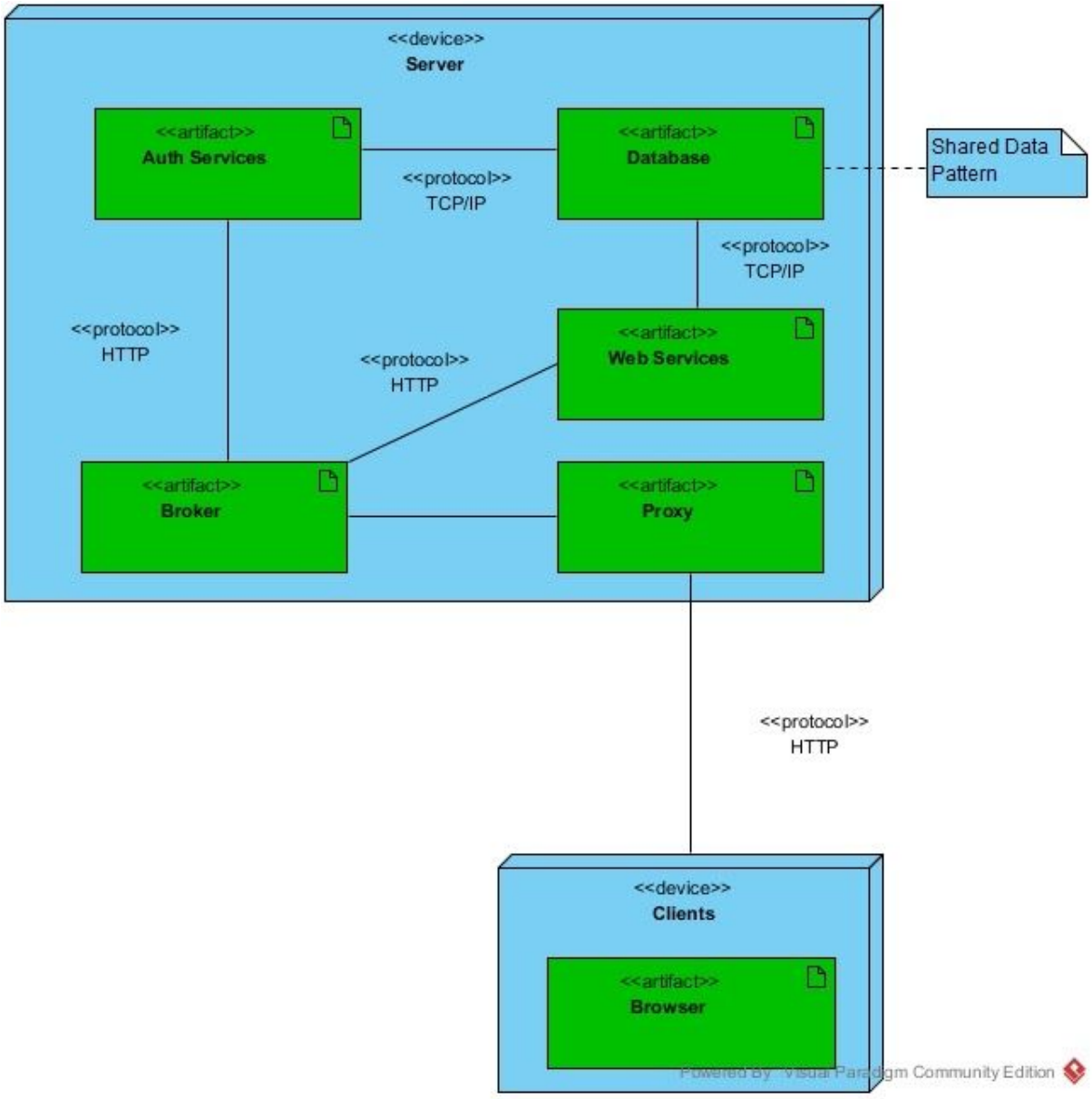
Elements

- Component: each lifeline represents a software component.

Relations

- Invocation [solid line arrow]: a component invokes the method of another component.
- Invocation response [dashed line arrow]: a component returns a response to an invocation.

Deployment Diagram



The system deployment onto hardware units revolves around the division imposed by the Client-Server and Broker patterns. As a cluster, those components can be allocated to the same hardware platform (as illustrated above) or alternatively distributed across multiple hardware platforms as demand increases. The broker, acting as the natural “entrance” to the cluster, is protected by a reverse proxy that serves to *limit exposure* as well as *limiting access* through the use of a firewall. The clients access the cluster through the Proxy-layer and reach the broker which then mediates the client’s request to the appropriate service(s). The clients run in a user’s browser and could therefore run on a variety of different hardware and software platforms, such as laptops, smart phones, and so on.

Element Catalogue

Elements

- Hardware platform: represents a hardware platform in the real world. Displayed through the <<device>>-tag.
- Software component: represents a software component that has been allocated to a hardware platform. Displayed through the <<artifact>>-tag.

Relations

- Allocated-to: a software component is allocated to a hardware platform. This is displayed through nesting.
- Communicates-by: a software component communicates with another component using a specific protocol. This is displayed through a line that specifies its communication protocol.

References

[1] Dangler, Jeremiah Y., "Categorization of Security Design Patterns" (2013).Electronic Theses and Dissertations.Paper 1119.https://dc.etsu.edu/etd/1119