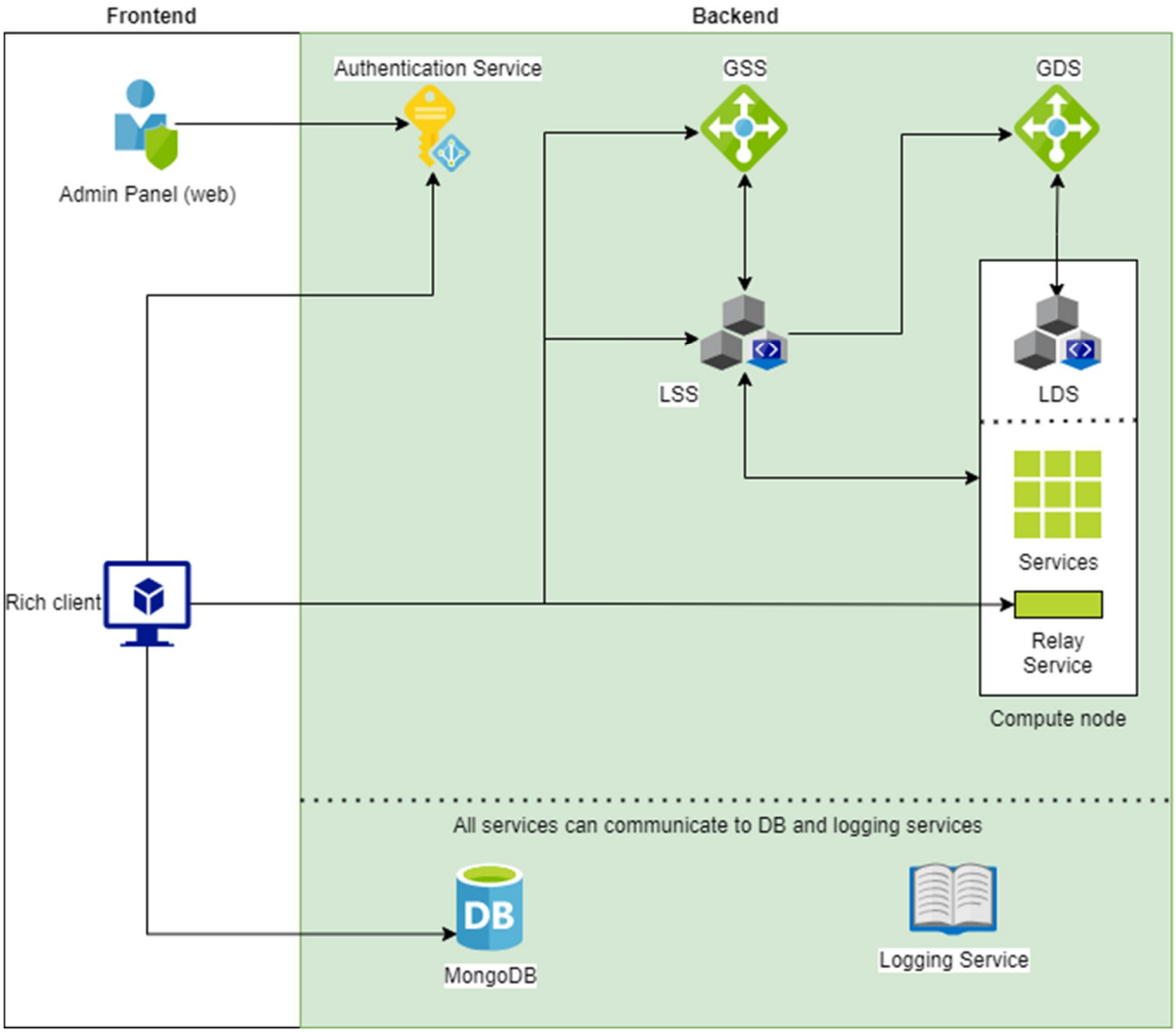
Open Twin Architecture Overview



Brief description of services

Admin Panel	The Admin Panel is a web application which allows an administrator to create and delete users in the system and manage the access of users and groups to projects.
Rich Client	The Rich Client is a local application which connects to the backend services and provides the user interface of the application.
Authentication Service	The Authentication Service manages the access control of the application. The rich client can send a request to check whether user credentials are valid. The Authentication Service is the only service which has admin privileges to the data base and controls the assignment of roles to users and groups.

GSS (Global Session Service)	The Global Session Service (short GSS) is a load balancer. It maintains a list of connected local session services and dispatches requests for creating new sessions from the rich client to one of the local session services. The purpose of the GSS is to manage the workload of the individual local session services. The workload of the GSS is very small such that it will be able to manage the requests coming from a large number of users. There should only be one GSS running in a system.
LSS (Local Session Service)	The Local Session Service (short LSS) manages a set of individual sessions. For each session, it controls which services need to be run. In addition, it controls and connects the services to each other. The workload of the LSS is moderate for each session such that each LSS can handle a number of sessions but may not scale up to a large number of parallel sessions. Therefore, the GSS acts as load balancer. When a new LSS is started, it will get the address of the GSS and register itself at the GSS.
GDS (Global Directory Service)	The Global Directory Service (short GDS) is a load balancer. Whenever a local session service needs to start a new service, it sends the request to the GDS. The GDS then determines on which compute node the new service should be executed (load balancing). Afterward the creation request is passed on to the corresponding local directory service. There should only be one GDS running in a system.
LDS (Local Directory Service)	The Local Directory Service (short LDS) actually runs and monitors services on a particular compute node. The LDS and the services it controls need to run on the same compute node, so there is one LDS per compute node. The LDS also maintains a list of services available at the particular compute node and communicates this to the GSS. The GSS will only distribute the request for available services to the corresponding LDS. When a new LDS is started, it will get the address of the GDS and register itself at the GDS. This allows for automatic scaling up and scaling down of the cluster setup.
Mongo DB	The MongoDB service is the central data base for the application. It also manages the access control to the project data. Each service may communicate directly to the data base service for performance reasons. MongoDB can scale horizontally by adding further shard or replication servers.
Logging Service	The logging service is a central service which logs activities or errors in the system. Each service can connect directly to the logging service and send log data with a time stamp such that the logging service can maintain the order of operations.
Services	This is a set of services where each service serves a particular purpose (microservices). The session type determines which services are started for a particular session. The services communicate via HTTP, so the services for a particular session may be distributed across a network. The services are started by the LSS through the GDS and LDS. User credentials are passed on to the services. This information is then used to connect to the data base for accessing the project data.

Relay Service	The Relay service is a special type of service which is managed similarly to the other services described above. The purpose of this service is to act as a relay between the Rich Client and the other services in the system. Since the Rich Client is expected to be located behind a firewall (with NAT), direct communication from the services to the rich client is usually not possible. Therefore, the Relay service has a bidirectional WebSocket connection to the Rich Client (established by the Rich Client) and is therefore able to send messages to the Rich Client. Any service which needs to send a message to the Rich Client can do this by sending it to the Relay service which then forwards the message to the Rich Client. In addition, the Rich Client only communicates to the Relay service which then forwards its messages to the corresponding individual services.

Communication between services:

- All communication between the services is performed via HTTP and encrypted by TLS.
- Communication from Frontend applications to services is protected by HTTPS.
- Communication between the backend services is protected by MTLS (Mutual TLS).
- Communication from all services to the database and the logging service is protected by HTTPS.
- Communication between the rich client and the relay service is partly performed through a WebSocket connection between the services. This communication is protected by TLS.

Activities for opening a new local client:

1. Users open the rich client on a local computer and logs in by specifying the address of the global session service (GSS) and provide credentials.
2. The frontend requests the address of the authentication service and the database service from the GSS.
3. The frontend sends the credentials to the authentication service. The authentication service responds whether the user is registered and active in the system.
4. If successful, the rich client opens and shows the list of projects being accessible to the user.

Activities for opening an existing project:

1. Users select a project from the list of projects.
2. The frontend sends the request for opening a new session to the GSS.
3. The GSS determines a suitable LSS (load balancing) and forwards the request to the GSS. The GSS then returns the address of the LSS to the frontend as response to the create session request.
4. The frontend then sends a message to the LSS for creating a new session and provides session information and the user credentials.
5. The LSS determines which services are required for the session type.
6. The LSS sends requests for starting the necessary services to the GDS.
7. The GDS determines which service should run on which compute node (considering load balancing and capabilities of the compute node) and sends the create service request to the corresponding LDS.

8. The LDS then locally creates the service on the compute node and provides the address of the LSS to the service.
9. After initialization, the service connects to the LSS and provides its address. The LSS therefore acts as a directory of all services in the session and can be connected by any service in the session.
10. Once all services have connected to the LSS, the LSS returns the creation message to the frontend and provides the address of the relay service. The frontend will then communicate to all services through the relay service only. The frontend also communicates to the database directly.
11. Once all services have connected to the LSS, the LSS sends the user credentials to all services, since this information is required for the services to access the data base.
12. The frontend then creates a WebSocket (bidirectional) connection to the relay service.
13. From here on, all services load the project information as needed by the individual services and start the interactive session creation process.

Activities for creating a new project:

1. Users selects the creation of the new project and specify the project type.
2. The frontend sends the request for creating a new session to the Authentication Service.
3. The Authentication Service creates the new project in the database and sets the access privileges accordingly.
4. From here on the following actions are identical to the opening of an existing project.

Authentication:

1. A list of users and credentials is maintained in MongoDB.
2. Each project has a project role assigned to is. This role allows assessing all assets of the project.
3. The authentication service creates the project collections (data containers) in the data base and also creates the corresponding roles.
4. If a user creates a new project the corresponding project role is automatically assigned to the user.
5. If groups of users are created, each of these groups also has a group role.
6. A group can have access to a project by assigned the project role to the group role.
7. A user can be part of a group by assigning the corresponding group role to the user.

In summary, a user can have project roles and group roles assigned. The user will then have to all projects belonging to the project roles being assigned to her/him. In addition, the user will also have access to all project for which their roles are assigned to a group being assigned to the user.

All group or role assignments for the projects are managed by the Authentication Service.

The access control is done at the data base level since every service connects to the data base by specifying the user credentials (username and password). The data base then determines via the role system whether the user has access to the project or not.