**Task 1 - initial design**

**Sub-task 1.1 - analyse requirements**

We need to implement a REST service on top of this model that provides CRUD operations, filtering, searching, sorting and aggregation. For this purpose we can use any framework.

But there are some requirements for db.

1. Data storage for REST service should provide support multi-tenant content storage.

It means that a single instance of a software application serves multiple customers.

In comparison to single-tenancy, multi-tenancy is:

* cheaper
* efficient resource usage
* lower maintenance cost
* larger computing capacity

1. Data Storage should scale so that:

 up to 100 000 000 of records are stored

 CRUD operations and typical search queries should execute within 1 second

1. Data storage should support automated data backup every 30 days.
2. Event Model for Data Storage Schema mapping complexity should be minimized. Based on this retirement its rather difficult to say what it means that complexity should be minimized
3. Data storage should support detailed logging for troubleshooting.
4. The infrastructure should support two European regions: east and west.

if your workloads require high availability and you have geographically distributed users who need low latency access to data, we are better off using a NoSQL database.

Sub-task 1.2 - define components

The application can be divided into to 3 layers.

* controller
* service
* repository

repository - design pattern that provides an abstraction layer between the application's data access logic and the underlying data source

service - business logic

controller - responsible for controlling the way that a user interacts with an application

Simple classes diagram

A screenshot of a computer

Description automatically generated

Sub-task 1.3 - define a flow

Use case “Send event”

A diagram of a data flow

Description automatically generated

Task 2 - DB choice

|  |  |  |
| --- | --- | --- |
| Title | Postgres | Mongo |
| Status |  |  |
| Background |  |  |
| Glossary | **Complex joins,**  **Foreign keys,**  **Transactions,**  **Large dataset handling,**  **Robust SQL support** | **Automatic sharding,**  **Flexible indexing,**  **Ad hoc queries,**  **Powerful aggregation framework,**  **Data validation,**  **Querying and indexing of JSON data** |
| Functional requirements | CRUD | CRUD |
| Non-functional requirements |  | Faster query times |
| Constraints | PostgreSQL, on the other hand, offers both Row-Level Security (RLS) and Column-Level Security (CLS), which restrict access to specific rows or columns within a table based on predefined security policies. | MongoDB offers Role-Based Access Control (RBAC), which restricts access to the database based on predefined roles, ACID. |
| Quality attributes |  |  |
| Solution Options |  |  |
| Decision criteria |  | up to 100 000 000 of records are stored  CRUD operations and typical search queries should execute within 1 second |
| Decision |  |  |
| Sources | https://www.postgresql.org/docs/ | https://www.mongodb.com/developer/products/mongodb/articles/ |
| Tickets |  |  |
| Reference |  |  |