TED UNIVERSITY

High-Level Design Report for EasySearch

Dinçer İnce

Halil Mert Güler

Süleyman Samed Çalışkan

Supervisor: Akhlaque Ahmad

Jury Members: Aslı Gençtav, Venera Adanova

1 INTRODUCTION

As the technology ecosystem grows with time, the amount of data stored around the world grows exponentially. Nowadays everywhere and everything around us produces or stores some kind of data, however data on its own is worth very little if it can't be analysed and be put into use. Natural language processing is an important field because of those reasons, it is a way to analyse and process written documents to be able to find the information we want. EasySearch is a system for doing exactly that, perform extensive search, analyse and process your documents without needing to learn or implement complicated natural language processing operations. EasySearch is an API where you can store your documents in folders called dictionaries. In those dictionaries you can perform automatic categorizations, perform extensive search based on the algorithm you can prefer, see various statistics about the dictionaries as well. EasySearch can be easily integrated to your application's stack or can be used as a standalone document management solution.

1.1 Purpose of the System

With EasySearch users can setup their configuration of the dictionary and documents, and through API they can access, add edit, and perform search operations on the documents without needing to implement any architecture. EasySearch is meant to be part of the user's application and instead of uploading their document in which they want to perform operations, they instead use the EasySearch system to upload the documents to EasySearch. The users then can have extensive search capabilities without needing to implement them, and drastically simplifying their implementation.

EasySearch can also be utilized as a standalone document management system as well, where users can store and organize their documents on the cloud, perform detailed search like a search engine, and categorize their documents automatically using machine learning. Although the main purpose of the system is meant to assist the user's own tech stack, its standalone value is also not to be underestimated.

1.2 Design Goals

The main purpose of EasySearch is, to search for any data or document in an instant time. In order to produce a working and expandable system we have decided to include the following properties in our software design:

1.2.1 Performance

Data or document uploading processes do not take long, and the most important process, searching in data, should be done in minimum time in accordance with EasySearch logic. Click actions within the site or API should be responsive.

1.2.2 Reliability

Users should not encounter errors on the EasySearch website or especially its application.

1.2.3 Completeness

The EasySearch application and website should come out with a complete version that can perform all the functions it promises.

1.2.4 Readability

After EasySearch's code is complete, it should be easy for anyone to read when viewed again.

1.2.5 Usability

EasySearch should be extremely easy to use and every user should be able to start using EasySearch quickly. If users see how easy and useful it is, more people will use the application.

1.3 Definitions, acronyms, and abbreviations

API: An application programming interface (API) is a way for two or more computer programs to communicate with each other. It is a type of software interface, offering a service to other pieces of software.

Dictionary: A folder like structure that stores and defines the type of the documents it holds.

Document: A type of object that stores a text and other relevant information defined by its dictionary.

Search: try to find something by looking or otherwise seeking carefully and thoroughly.

Server-side applications: Server-side apps are the most common type of application encountered when dealing with OAuth servers. These apps run on a web server where the source code of the application is not available to the public, so they can maintain the confidentiality of their client secret

1.4 Overview

EasySearch is an API and website where you can store your data and documents and perform detailed searches on them. The fact that EasySearch is both an API and a website is up to the user's preference in terms of usage, but it is recommended to use both together.

EasySearch can store and store your documents and show the statistics you want on them and make the searches you want. You can also folder and categorize all your documents as well as edit them.

With EasySearch, users can easily learn and practice how to quickly search their documents and data with the simple use of EasySearch.

2 Proposed Software Architecture

2.1 Overview

The architecture of the EasySearch system is explained in the sections below. The overall system design that would fit the prepared analysis report including the subsystems, modules, interactions, and technologies used in them is discussed.

2.2 Subsystem Decomposition

The architecture of the system is designed in a way to be as flexible as possible to be able to fit to any scenarios a user might require. The importance of flexibility is crucial in such a system as the system is responsible for holding the data of the user, as such the user shall be able to interact with the system in different ways as well as store their data with as many different configurations as possible.

There are two ways to utilize the system, which is from the API, or from the web application. The web application allows the user to access all the functionality of the system whereas the API is only used for the document interactions.

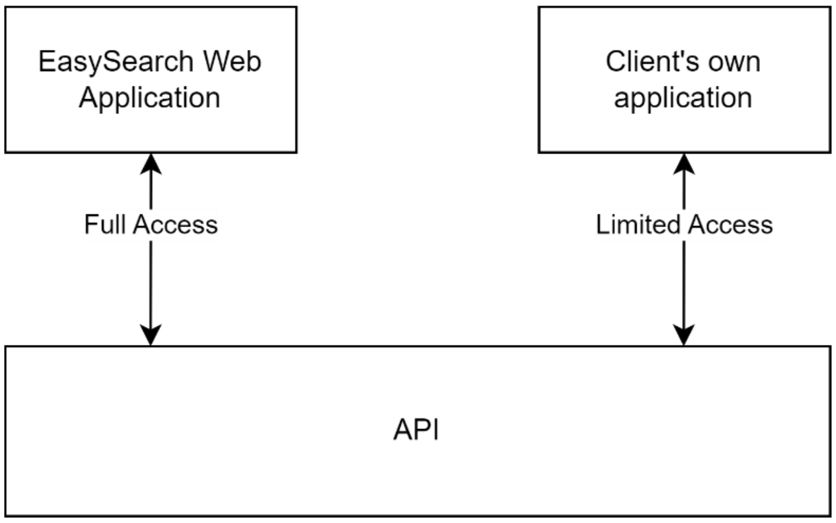

Figure 1 Access Limitations

The main reason for this architecture's design choice is due to security considerations, the web application allows the creation of API keys, as well as the API's implementation configurations, with this separation the user can configure the API for their specific use in their application, completely from the web application.

2.2.1 API

The API side of the system is separated into four main subsystems, authentication, request handling, repositories, and services.

Request handling subsystem is the first subsystem that interacts with the incoming request, it forwards the request to the authentication subsystem to check the validity of the request, then to the responsible repository, then gives a response.

Authentication subsystem handles two different tasks depending on the type of request it receives. For requests concerning, user state (ex: login) and dictionary management, it checks the validity or creates a new JSON Web Token. These types of requests should come from the EasySearch Web Application as they concern the management of the API. For requests concerning the document management, it checks and validates the API key given in the request. The subsystem verifies the API key and if the permissions of the API key allow it access for the dictionary the document is stored in, it allows or denies access.

Repositories is the subsystem that interacts with the database, and responsible for handling the data. After the request is sent to the responsible repository, repository can fetch the necessary data, store, or make changes to the requested data, or can process the data with the help of the services before storing or sending back the data.

Services is the subsystem that handles most of the processing applications of the search engine, it handles tasks such as word processing, categorization of the documents etc.

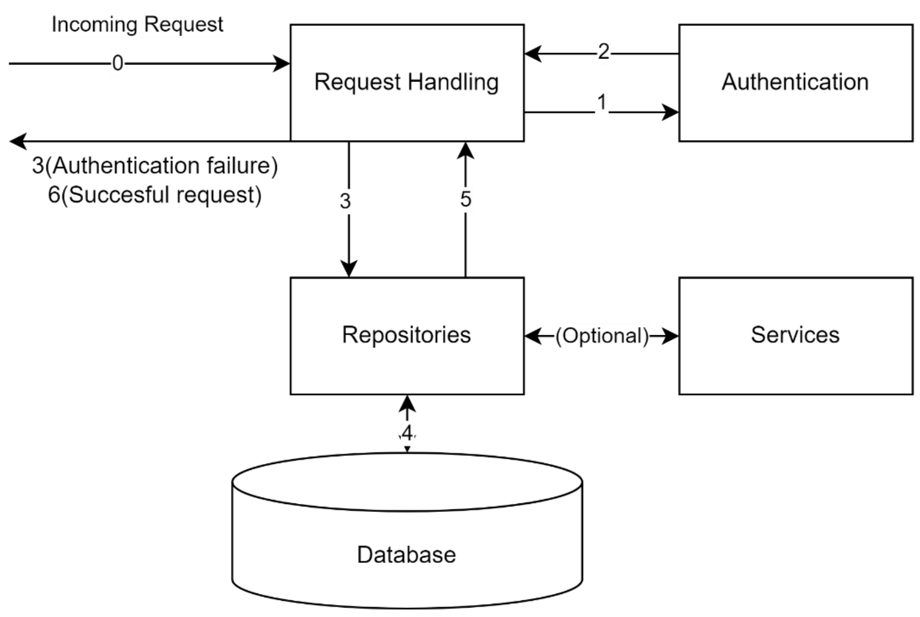

Figure 2 Interactions between the subsytems, numbers on the arrows represent the order of the execution

An Incoming search request's subsystem interaction would be as follows, first in the request handling it would check the validity of the request (valid endpoint, correct parameters etc.), then it would be forwarded to the authentication to check the API key of the request. If the API key is valid and has permission to the dictionary the search request is asking for, if the authentication is successful, the request would be forwarded to the responsible repository in this case the documents repository, in the documents repository the search query would be sent to the preprocessing service to be processed to become a document, its words would be added to the inverted index, then according to the dictionary's preferred algorithm it would be sent to the searching service where the search operation would be performed, then the response would back trace until the request handler where the response would be sent to the user.

The API's main use case is to be utilized by the client's application whether it is a desktop, mobile, or web application as long as it can communicate to the API through HTTP requests the API can be used. The system's main job is not to replace the backend of the client's application but only to be used for the documents the user want to perform search or categorization operations on.

In the case that the user would only want to utilize the system as a document management solution for personal use, then the system can be utilized solely through the web application.

2.2.2 Web Application

The web application side of the system is separated into three main subsystems which are, user management, API management, and document management.

User management subsystem handles the tasks related to the creation and configuration of the user profiles. All actions related to the personal information of user is handled by this subsystem.

API management subsystem allows the users to manage the dictionaries, such as the creation of the dictionary, configuration of the dictionary in terms of the additional information the documents will hold (extra columns to store the data for the document), the search algorithm the dictionary will use, and handle the categorization of the documents inside the dictionary, as well as see various statistics about the usage of the API and observe the logs.

Document management subsystem allows users to directly manage their documents. The users can perform search on the dictionary or the documents. Add, delete or edit their documents completely from the application without the need for a separate client application. This subsystem is for the users that doesn’t want to use the system as a part of their application stack, but to utilize it standalone.

2.3 Hardware/Software Mapping

The system’s API side will be centralized and will be running in a single server, along with the database. The API will be implemented in c#, running the .NET 7 framework. If the load starts to affect the system’s performance, the database can be distributed with replication to balance the load.

The web application can be run on any device that supports modern browsers (ex: chrome), including desktop, mobile or tablet, and has an internet connection. The web application will communicate with the API through HTTP requests.

The optional client's application will also communicate with the API through the HTTP requests; however, the API will be limited in that case to document management operations. Both the web application and client's application will allow the document management operations, such as adding, deleting, editing documents, and extensive search functionalities to find such documents.

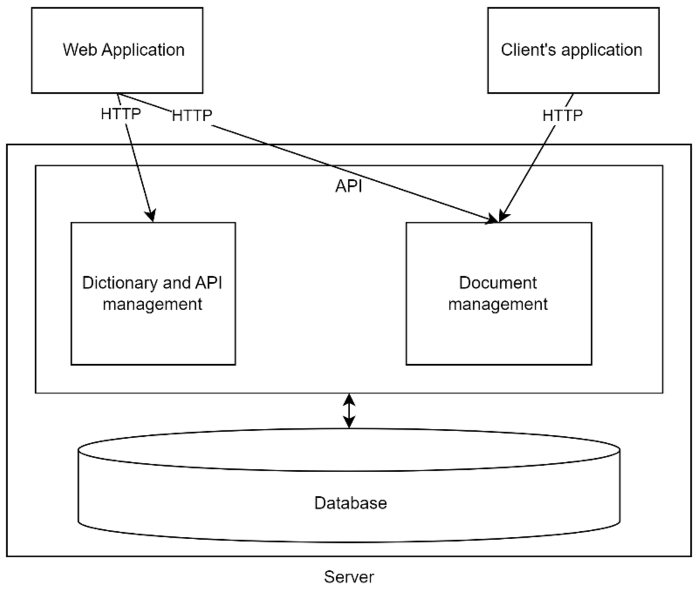

Figure 3 Hardware and Software Interactions

2.4 Persistent Data Management

Persistent data includes the user data, dictionary data, and document data. All the persistent data is stored in the servers of the system. For the data storage we have chosen a relational database as the type of data store is highly relational and the relation by ID's improves the speed of the queries dramatically compared to a NoSQL solution. For the relational database we have chosen PostgreSQL as it has proven to be reliable and fast over the years to be consistently used by big technology companies, as well as it is being free and open source. For the mapping of the data to the API we are using an object-relational framework, specifically entity framework, to have relatively easy queries, and simpler mapping of a data to an object.

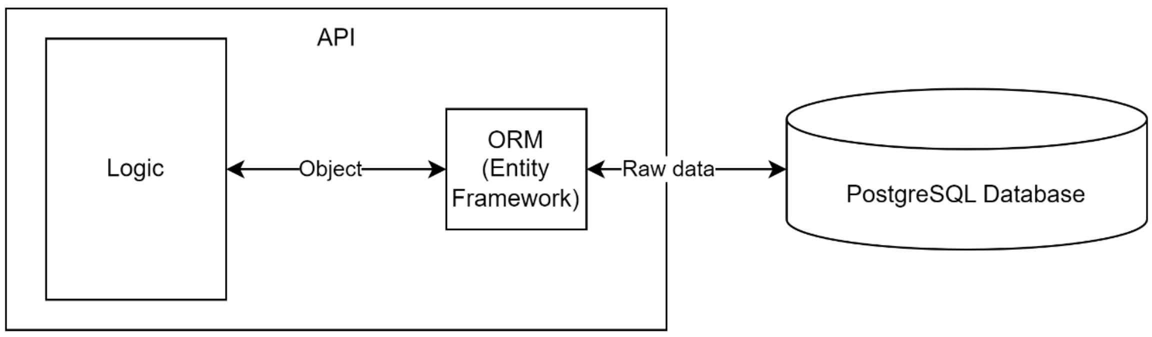

Figure 4 API's data management and storage

The user data is mainly used for authentication purposes, both for the management of the API for the web application via tokens, or the usage of the API through the API keys. Dictionary stores information similar to a metadata of a file, defining the type of the document it is storing. Document data are used for searching for fetching the documents.

2.5 Access control and security

Security is one of the most important aspects of the application. The application contains all the storage system information of the user. Data such as the number of documents, their content will arise. The data in the system should not be leaked in any way. For this reason, we have put in multiple security measures as access controllers.

The project uses two levels of access control. First one is for API control panel and second one is API key.

API control panel is a web application that uses model-view-controller architecture. When user want to use the application its necessary for user to access API control panel. For this scenario, user needs to register first, after that when user login to API control panel, a unique token will be generated for that user, and it will provide access to API control request for a certain time. For this toke, the app will use Jwt that provided by Bearer.

API key is necessary for user to use applications API requests, and every user has unique API key for themselves. With this key, users can access to general API request for their application.

The access matrix:

Actors/Object	App	Api Requests	Api Control Panel
Admin	Logs and Data flows	-	-
Application User	-	CreateAPI Requests	View/Update Documents and Dictionaries

2.6 Global software control

The application has one main service, but it will hold both API requests and API control panel request. Event-driven control shall be maintained for both the application server as well as the control servers.

- Each sub-service's workflow will be independent.
- When user create dictionary, it will directly send request to the main application.
- Created dictionaries and documents can be controlled by user from API control panel.
- Search sub-service will read created dictionary as soon as it created.
- These two sub-services will work simultaneously.

2.7 Boundary conditions

The application has boundary conditions in order to provide functionalities correctly. These boundary conditions are significant for developers and users, they use these boundaries for examining each component and analyzing the failures. Consequently, these conditions let developers to handle the disorders. In other words, boundary conditions state the initial behavior of the system when it’s starting up, shutting down and even in error conditions.

2.7.1 Initialization

At first, users need to API web application to use EasySearch API. API web application can be reach from any web browser. In API web application firstly, user need to register EasySearch to be able to use configurations and the API. Account creation can be initiated by the related button on the landing page. If the login credentials do not match with user information, authentication fails. The user is redirected to the login screen. Since the authentication and log operations takes place, internet connection is required at this stage. When the user logs in to API web application, user will be redirected to dashboard. The authentication token is saved to cookies on web browser, so the user can access the account throughout cookies are exists without authenticating again.

2.7.2 Termination

The user can log out of API web application at any desired time, however this operation does not effects working procedure of the main API. It only ends user session which created on API web application. Also log out operation will be reset and clear cache and cookies in the web browser for application web site so user will need to be log in in order to reach the API web application.

2.7.3 Failure

In cases of failures such as connection lost, instant exit or hardware related issues, system will complete created request/requests, if it's sent already before failure. If request/requests created but couldn't send to server, information related to them will be lost as well.

3 Subsystem services

The project is a server application, so it provides API, but it has an API web application which is a client-server application. Both API and API control panel have same server. Client side of the application is basically frontend of API web application. The client contains end point services and user interface for configure and manage the main API's features. It has a component base design for every page. On the other hand, end points will be held in services.

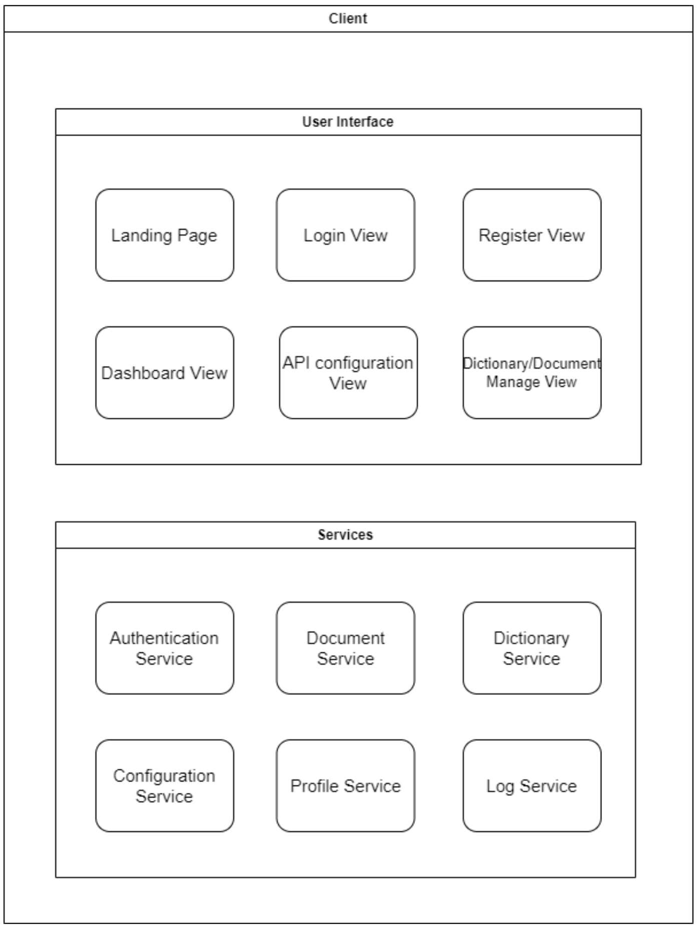

Figure 5 Hierarchy of modules inside the client system

3.1 User Interface

- Landing page: The view of API landing page that given information about the EasySearch API. Also, login and register buttons are on this view.
- Login view: The view of the login page to authenticate users into API web application.
- Register view: The view of register page to create new user of EasySearch.

- Dashboard view: The view that shows general information about the API situation. Basically, list of dictionaries, documents search algorithm and other information that will be necessary for user.
- API configuration view: The view that allows to select the search algorithm for the main API.
- Dictionary/Document manage view: The view that allows to list, create, update and delete dictionary and document.

3.2 Services

- Authentication service: Service that provides endpoint for log in and log out operations. Also guard and interceptor components for security will be hold in here.
- Document service: Service that provides endpoint for operations such as get, create, delete, edit documents.
- Dictionary service: Service that provides endpoint for operations such as get, create, delete, edit dictionary.
- Configuration service: Service that provides endpoint for API configurations.
- Profile service: Service that provides endpoint for operations such as get, delete, edit user information.
- Log service: A background service to track user behavior on API web application.

Server side of the application includes both main API and API control panel entities. Server side holds database context, models, data transfer objects, repositories, and controllers. Server side works with API key and token that created for user in case of security. Main purpose of server side is creating database injectable queries and collect or change the data. And after that sending them to the client by using endpoints.

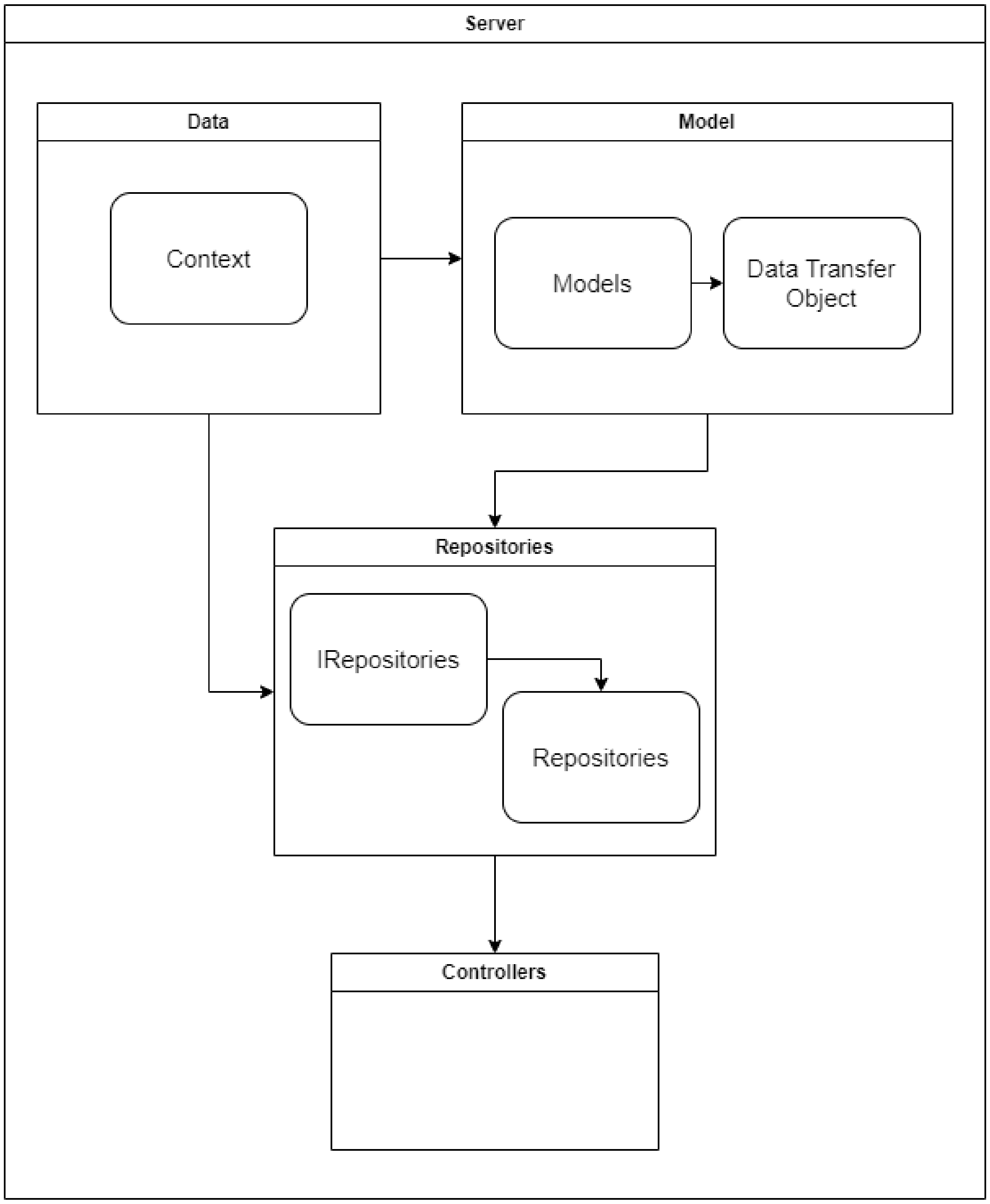

Figure 6 Hierarchy of layers and submodules within the server

3.3 Data

- Context: Database context created with Entity Framework. Connects queryable operations to database.

3.4 Model

- Models: Collection of objects of the system created by responding tables and columns in the database.
- Data transfer objects: Same with model but can include external variables different than the corresponding model object.

3.5 Repositories

- IRepositories: Interface classes of repositories to implement on and use on the corresponding repository and controller.
- Repositories: Level of data operations, queries and calculations. Generated data can be transmitted to controllers by interface of the corresponding repository.

3.6 Controllers

- Collection of HTTP requests generated by the data came from repositories in order to create endpoints.

4 Glossary

- API: An application programming interface is a way for two or more computer programs to communicate with each other.

- Request: A request includes the URL of the API endpoint and an HTTP request method.

- Dictionary: Object type that created by user holds document and implements selected search algorithms to documents on it.

- Document: Created documents by user, search algorithm will be work for documents and their contents.

- Jwt: JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims.

- Bearer: Bearer Tokens are the predominant type of access token used with OAuth 2.0. A Bearer Token is an opaque string, not intended to have any meaning to clients using it.

- ORM: We will use for managing to data base. Object–relational mapping in computer science is a programming technique for converting data between type systems using object-oriented programming languages

- Interface: type definition similar to a class, except that it purely represents a contract between an object and its user

- HTTP requests: HTTP requests are messages sent by the client to initiate an action on the server.