Lendsqr API

Introduction

This introduction provides an overview of the Lendsqr API. It includes the purpose, scope, target audience, design approach and design patterns of the system.

Document Scope and Purpose

This document provides a description of the technical design for the Lendsqr API. This document's primary purpose is to describe the technical vision for how the requirements will be realized. This document provides an architectural overview of the system to depict different aspects of the system. This document also functions as a foundational reference point for developers.

Target Audience

This document is targeted (but not limited) to technical stakeholders:

- Development Team
- IT Management
- Support Staff

It is assumed that the reader has a technical background in software design and development.

Design Approach

The design approach used here is based on the following:

Data Flow Design

The data flow of the API can either be cloud based or local. Data flows from whichever MySQL provider is used to and from the application.

Architecture Design

The API will follow a Six Layer Architecture so that the objects in the system as a whole can be organized to best separate concerns and prepare for distribution and reuse. A principal advantage to this design is the relative stability of the components as seen by the applications developer. Implementations may change considerably to enhance the performance or in response to changes in the architecture. These changes are less likely to cause major impact to the applications' programs.

Design Patterns

This application is designed using a six-layer architecture by factoring with the following layers:

The Controllers layer: This layer contains all the controller functions which help the application to work as expected. It contains all the logic for the application. The files in this layer are:
AUTH Controller – To control authentication across the application
Error Controller – Controls the global error handling of this application
User Controller – Contains al the logic for tasks related to the user resource

The DB Configuration Layer: This layer contains everything relating to the data base schema, migration and connection settings. The files and folders in this layer are:
Models - A folder which holds all the migrations to be carried out on the database
KnexFile – A file which contains configuration settings for the KnexJS ORM based on different environments
DB – Configures the database based on the current environment and configuration settings in the KnexFile

The Routes layer: This layer contains the routing of the entire application. Files in this layer are:
The user router – Controls all routes related to the user resource as well as user authentication.

The Utilities Layer: This layer contains helper functions which are never used in the application directly but leveraged in the controllers.

The Main Application Layer: This is the main layer of the application which controls the running of the application itself. Without this layer, the application would not even start. Files and folders in this layer are:
The app file – Initializes the application

The server file – The application is actually started in this file
The config helper file – File to help environment variables reach inaccessible locations
Node Modules Folder – Contains all the libraries and code for the tools needed for this application to work
The tests folder – Contains all the test for this folder
Gitignore file – Lists the sensitive files or folders that should not be uploaded to git
Package JSON files – Contains the app information and metadata
Readme File – A little documentation is included in this file to help understand the project

The Data Layer: This layer is managed solely by the Database(MySQL)

Authentication

Authentication of this application is token based and this was done harnessing the JWT(JSON Web Token) concept/library on Node.js.

Usage

This API should be consumed from a client using a URL specified in production. To consume this application locally, an API client like Postman or ThunderClient can be harnessed. The application should be cloned from GitHub and the server should be started using the node server command in the root folder. Please ensure that Node.js is installed on the machine to be used.

Environment Variables

A config.env file should be created in the root folder before starting the server and all the necessary environment variables should be added as in code. Environment variables needed in the file are as follows:

- **PORT –** The port the server should listen on
- **ENV –** The environment of the project, should either be “development” or “production”
- **DB_DEV –** The name of the development database
- **DB_DEV_PORT –** The port the development database runs on
- **DB_DEV_HOST –** The host the development database runs on
- **DB_DEV_USER –** The user granted permission for the development database
- **DB_DEV_PASSWORD –** The password for the development database
- **DB_PROD-** Should be a connection string for the production database

- **JWT_SECRET –** Should be a long string to be used by JWT
- **JWT_EXPIRES_IN –** The amount of time the JWT lasts (for 90 days it should be set to 90d)
- **SESSION_SECRET –** Another long string

Connection and consumption

Once the development server has been spun, various endpoints can be visited using localhost domain and the port passed in the configuration file.
Here is a documentation for the API consumption
This documentation contains the full list of routes and endpoints to be used in this application. The URL in the documentation is based on port 8000.
It also explains the request and response structures in-depth.

Tools and Technologies Used

- **NodeJS – Node.**js is the platform(language) used to run this entire application
- **ExpressJS –** This is the library built upon node.js to build the API.
- **KnexJS ORM –** The SQL query builder for JavaScript used.
- **MySQL –** The database used for the application

A full list of tools and technologies used can be found in the package.json file

GitHub Repo