Sticky notes application

Mikita Akulich

https://github.com/nick-ak96/sticky_notes

Overview

The “Sticky notes” application is an account-based web application. It allows users to create and manage sticky notes as well as share them with other users, user groups or the whole world.

Content structure

The content of the application can be logically divided into two main parts Home page and User account page.

Home page

The Home page is the first page a user sees when accessing the application. This page is accessible to users that have an account as well as unauthorized users.
On the Home page users are able to

- Create a new account
- Log in into an existing account
- View public sticky notes

Account creation

An unauthorized user can create an account. User account has a unique username, password, name and surname. A user can edit account password as well as name and surname. Since an account username is unique changing of username is not allowed.

Log in into account

An unauthorized user may login to an existing account with his credentials such as username and password. The application does not have a feature for password reset, so the password must be remembered by the user.

Viewing public notes

Public notes are arranged into a dashboard on the Home page. The notes are sorted by the creation date to form a timeline with most recent notes at the top of the dashboard. Editing of notes on the Home page is not allowed even for the note owners. The notes can be searched by a username or content with the search box at the top.

User account page

When user wants to actually user the application, he must navigate to his account page.
There a user has four different components:

- User notes dashboard
- Shared notes dashboard
- Organizations
- Account settings

A user can navigate to each of the components with the navigation menu at the top of the page.

User notes dashboard

The User notes dashboard consists of notes that the account holder has created. A user can view, create, edit, share and delete his notes from this dashboard.
The notes are sorted by creation date and can be filtered by content using the search box at the top of the dashboard.

A new empty note with default color is created with the click of a button "Create note". A note at this moment is private. The user can edit note's content, color and make changes to its sharing properties, i.e. share a note with an existing user, share a note with organization that the user belongs to.

During note shared the degree of sharing can be specified. A note can be shared with 'read-only' or 'write' access level. Of course, a note shared to public is 'read-only' by default which cannot be changed.

Shared notes dashboard

This dashboard is similar to the User notes dashboard, which is composed of notes that other users have shared with the account holder. A note here may be editable if the note creator shared it with the 'write' access level. Note, the deletion of a note is permitted only to the note owner and nobody else.

Organizations

The Organizations component is composed of organizations where the user is a member. A user can create, edit and delete organizations. Each organization must have a name, creator and a list of members. An organization can be edited or deleted only by its creator.

With a list of organizations presented a user may navigate into an organization. Organization view will contain a dashboard with shared notes and a list of members. When a user is added to an organization, he can find a new organization entry in his account.

Notes in organization are added or removed with sharing. That means that in order to create a note in an organization, a user must create a private note in User notes dashboard and then share it to an organization. In order to remove a note, a user simply needs to remove the organization from sharing list. All the notes in the organization view are ‘read-only’.

Account settings
In Account settings view a user can edit account password as well as his name and surname.

Application architecture
The application is implemented using Model-View-Controller architecture and has a layered structure. The layers of the application are as follows.

- Model
    - Business logic layer
    - Data Access layer
- View
    - Presentation layer
- Controller
    - Web API layer

View
Presentation layer of the application is obviously a web-based user interface. This layer is implemented using **Vue.js** framework which utilizes plain JavaScript, CSS and HTML languages.

This framework was chosen because of its simplicity and capability. It uses plain JavaScript, enforces code reusability, offers a reactivity mechanism and has a very competitive performance score.

This layer communicates with the API using AJAX requests.

Controller and Model
The Web API layer and other back-end layers are all implemented using .NET Core framework in C# language.

This framework for server side was chosen because it is cross platform and powerful. Also, it integrates with the OpenAPI specification standard and utilizes some very nice design patterns that will be discussed later.

Controller

Authentication and authorization
The Web API uses token-based authentication, in particular JSON Web Token (JWT) standard. The token signature or hash is computed using the HS256 algorithm to ensure token integrity and authenticity.

The token is issued upon user authentication by username and password and is stored on the client. Each following request is authorized by verifying the Authorization header of the request on the server. The Authorization header must be of Bearer type. The token has expiration date and is valid for as long as 1 hour from creation.

Additional layer of authorization mechanism is implemented by encoding a user ID into token payload upon its issue. This allows to control user access to certain actions and verify if the user is authorized to perform them. As an example, the deletion of notes and organizations utilize this mechanism to check whether the user is certainly the owner of an entity to perform its deletion.

Content negotiation

The Web API can provide only JSON representation of resources in responses.

API

The Web API consists of the following controllers:

- User
    - CRUD and authentication
- Note
    - Accumulation of public notes
- NoteSharing
    - Sharing of notes from owner to other users or organizations
- Organization
    - CRUD
- OrganizationUser
    - Controls membership of users in organizations
- UserNote
    - CRUD of private user notes
- OrganizationNote
    - Accumulation of organization notes

Model

The business logic and data access layer are implemented using dependency injection technique. This allows granularity of services, their scalability and easy development of new functionality as well as hot swapping of application components.

Storage of sensitive data

The password storage is obfuscated by using salted hash SHA256 of a password. The salt value together with the hash are stored in the database upon account creation.

Database

Persistent storage is represented by SQLite database. As user base scales, the storage representation can be easily migrated to another database that users SQL.