3. Core of the work

In aiming to create a stable and dependable solution, combining modern and mature technologies was necessary. In order for the mix between the two to be successful and not lead to unexpected issues, the structural and architectural layout of the application had to be very well defined, be it the separation of projects within the solution through an N-Layered architecture or the structure of the database.

Text

Description automatically generated In order to preserve principles and best practices necessary for a SOLID programming environment, the application was structured as for allowing ease of use and effortless scalability through popular and common separation and abstraction techniques. Moreover, tests were implemented in order to ensure a normal and predictable behavior of the features within the application. Endpoints and base methods for more used features were thoroughly tested. Moreover, natural flow and responsiveness, among others, were tested on the UI through popular web testing framework Selenium.

Figure 3 - Projects' layout within the solution

Graphical user interface, application

Description automatically generated3.1 Problem and solution

Figure 4 - Tenant change menu

The aforementioned problem of the current situation was that information could not be accessed by users or merchants and the process of gaining access to that information was tedious and frustrating. Through the developed solution, any used can access all the information they might need directly from one source, albeit only by creating an account. Not only that, but merchants can also deliver their products to clients by simply having a Tenant Account created. Once that is done, company users can manage the items for sale, client orders and check statistics.

Graphical user interface, text, application, email

Description automatically generatedAfter logging in, one of the first features available for every user is the Events section.Graphical user interface, application, Teams

Description automatically generated In this section, administrator can modify or delete existing events or create new ones. Here, users can see past, ongoing or future events in a table featuring pagination and filtering while also being able to only select specific event types from a drop-down menu. This behavior is implemented through jQuery DataTables and permissions for the administration of this sections is managed through permission checkers in the page’s code. Such permissions can be managed by administrators in the Roles section of the application, allowing for different roles to have specific permissions assigned to them, without the need for roles to be made for each task. Each user can have a role, and each role can have different permissions, enabling administrators to easily manage activities through configurable settings. On tenant creation, an admin account is automatically created and given all permissions. However, there can also be host-specific permissions awarded only by the host administrator, such as tenant management. In code, this is achieved through an AuthorizationProvider class that can be configured for the needs of the application.Text

Description automatically generated By specifying the tenancy side that can obtain a permission, the permission only becomes available to host accounts or tenant accounts, whichever is specified. This type of behavior allows for further configurability and separates host admins from tenant admins without the need for complex permission checks.

Figure 5 - Edit role modal

Figure 6 - Events section

Figure 7 - Authorization Provider for permission creation (Code snippet)

Text

Description automatically generated Events can be accessed both by admins as well as regular users, yet can only be managed by admins which have been granted the Events Management permission. This is only done through checking permissions in the front-end, meaning that in the back-end the endpoints are not permission-proofed.

Figure 8 - Event Application Service (Code snippet)