# PetShopDemo WebAPI

### Description

The PetShopDemo is a web Api application created as part of the BallastLane selection process. It was created to demonstrate the skills and best practices developing modern Web API applications and general Software Engineering.

The tools used to create the Web API are:

* .Net 8 with csharp programming language
* Microsoft Azure Table storage
* Postman for Web API testing
* Visual Studio as IDE
* StarUML for modeling
* Github as source control tool

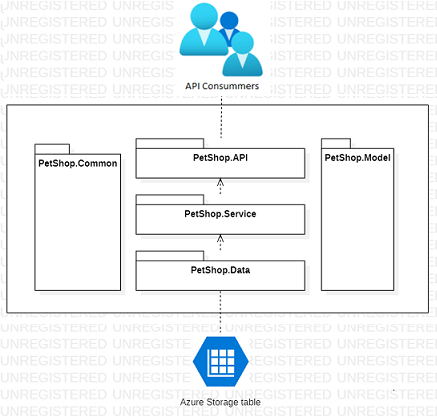
Third Party libraries used in the development process are:

* FluentValidation for API endpoints validations
* Serilogs for logging
* Moq for mocking unit tests
* Nunit for unit testing
* FluentAssertions for unit testing validations

Concepts applied:

* RESTfull web services
* JSON Web Token Authentication
* Password Hashing
* Unit Testing
* Layered architectures pattern

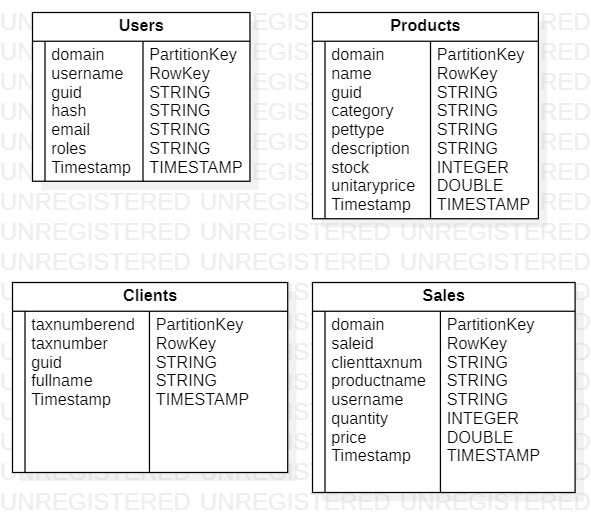
#### Package Diagram:



#### Azure Table Storage

For fast development and easy deployment, the Azure table storage was selected as data repository. Then, the PetShopDemo WebAPI requires access to an Azure Table Storage configured to use the DefaultAzureCredentials. See more details in the Deployment section

The PetShopDemo WebAPI uses four tables: *Users, Products, Clients, Sales*. These tables can be created using the ‘*[POST] api/Setup/’* endpoint provided by the Web API. See more details in the Deployment section.



### Deployment

The PetShopDemo Web API code can be found in the following Github repository:

* <https://github.com/josehvaldes/PetshopDemo.git>

You can clone the repository or import it into another Github account so that it can be access from you Azure Portal account.

Create a new ‘**Web App** **Service**’ in Microsoft Azure and use the PetShopDemo repository and the ‘main’ branch. Since the PetShopDemo API uses the “DefaultAzureCredential”, configure the new ‘**Web App Service’** to use a ‘*System Assigned’* Identity or a ‘*User assigned identity’*.

Create a new Storage Account and use the ‘Access Control (AIM)’ option to assign the *‘Storage Table Data Contributor’* role to the managed Identity created in the previous paragraph.

Finally, update the following key in the new **Web App** settings with the URI of the new azure Storage Account:

Key = ***azureSettings.StorageURI***

Value= *“https://{your\_storage\_account}.table.core.windows.net/”*

### Create Tables

Assuming that the storage account is empty, use the following endpoint to setup the needed administration user and mockup data:

* [POST] api/setup/

The endpoint will create the needed tables and add default/testing data. If the table already exists, the data will not be added.

The default administrator account credentials are:

*Username: admin*

*Domain: bo*

*Password: 123456*

### Postman

Now you can import the “PetshopDemo.postman\_collection.json” file included in the source code in ‘Postman app’ to start testing the Web API endpoints.

You can check the endpoints, parameters and responses by loading the “PetShopDemo\_swagger.yml” file in a YAML viewer like: <https://editor.swagger.io/>

### Usage

The PetShopDemo ‘*Users’* and ‘*Products’* and *‘Sales’* are organized logically in “domains”. A domain can be a city, country, or store. The Domain is the PartitionKey used by Azure table storage. The clients table are partitioned using the last number of their ‘*tax numbers*’ so that clients are distributed equally. This was a design choice.

The PetShopDemo WebAPI provides the following uses cases:

#### ‘User’ Use cases

* As an operator, I want to login using the default Administration account
  + [POST] [api/users/login]
  + The post action returns an authentication token that will be used to access the other endpoints.
* As an operator, I want to create a new user to login.
  + [POST] api/users/
  + Restrictions:
    - New users must have the ‘User’ role only.
    - Only a user with Administration Role can create new users.
    - A user’s username must be unique in its domain.
* As an operator, I want to delete a user with non-administration role
  + [DELETE] api/users/{domain}/{username}
  + Only users with Administration role can delete other users
  + Users with the Administration role cannot be deleted
* As an operator, I want to retrieve information of a user
  + [GET] api/users/{domain}/{username}
  + Only users with Administration role can execute this operation.

#### ‘Product’ Use cases

* As an operator, I want to create a product
  + [POST] api/products/
* As an operator, I want retrieve a product with specific domain and type
  + [GET] api/products/{domain}/{pettype}?availablesOnly=false
  + The [availablesOnly = true] query parameter will be used to recover products with stock higher than zero. Use false to include products with stock equals to zero
* As an operator, I want to update a product
  + [PUT] https://localhost:7025/api/products/{domain}/{name}
  + Name and Domain properties can’t be updated.
* As an operator, I want to delete a product
  + [DELETE] api/products/{domain}/{name}/

#### ‘Client’ Use cases

* As an operator, I want to create a client
  + [POST] api/clients/
* As an operator, I want to retrieve a client
  + [GET] api/clients/{taxnumber}
* As an operator, I want to update the name of a client
  + [PUT] api/clients/{taxnumber}
  + The fullname is the only property to be updated
* As an operator, I want to delete a client
  + [DELETE] api/clients/{taxnumber}
* As an operator, I want to create a sale
  + [POST] api/clients/
* As an operator, I want to retrieve sales
  + [GET] api/clients/?domain={domain}
  + Only sales of a specific {domain} can be retrieved

#### Sales Use case

The Create Sales endpoint requires additional explanation:

* The domain and username of the Sales table is the same domain as the authenticated user. This information comes from the HTTP context.
* If the Client’s tax number sent in the request doesn’t exists, a new client is created.
* If the Client’s tax number exists, the client’s fullname is compared with the client in the repository. If the names don’t match, the sale is cancelled.
* If the product’s stock is lower than the request’s quantity, the sale is cancelled. To complete a sale the product needs to have available stock.
* If the request price doesn’t match with the expected price: (product.UnitaryPrice\*request.quantity), the sale is cancelled.
* The product’s stock must be updated and discounted after the sale. If the update process fails, the sale is rollbacked.

### Unit Test

Because of the time constrains only the PetShopDemo.Service library has unit tests. The unit tests can be found in the *‘PetShopt.Tests’* library