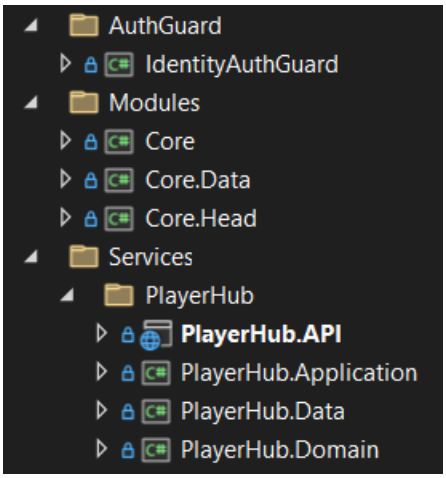
Integrator Project

Project Dependencies:

- .NET Core v8.0.11
- ASP.NET Core v8.0.11
- .NET Standard Library v2.1.0
- Microsoft SQL Server 2012 or later

Structure:



MODULES: Contains all the requirements to start to develop any application.

- **Core** (domain layer requirements using .NET Standard Library v2.1.0): base Model, base DTO, base Domain Event.
- **Core.Data** (data access layer requirements): base DbContextFactory, base Repository, base Seed (populating a database with initial data), base UnitOfWork.
- **Core.Head** (application layer requirements): Behaviors (create pipeline behaviors to handle logging and validation to intercept and process commands, queries and events), base Command and CommandHandler, Query and QueryHandler, custom Exceptions and Global Exception Handler, APIs response Wrappers.

AUTH GUARDS: Provides a tailored solution for Authentication and Authorization within the application.

AUTHGUARDIDENTITY:

- Custom implementation of identity model mixed with custom implementation of JwtBearer.
- Provide the way to register a user, login, logout using a JwtBearer and refresh tokens.

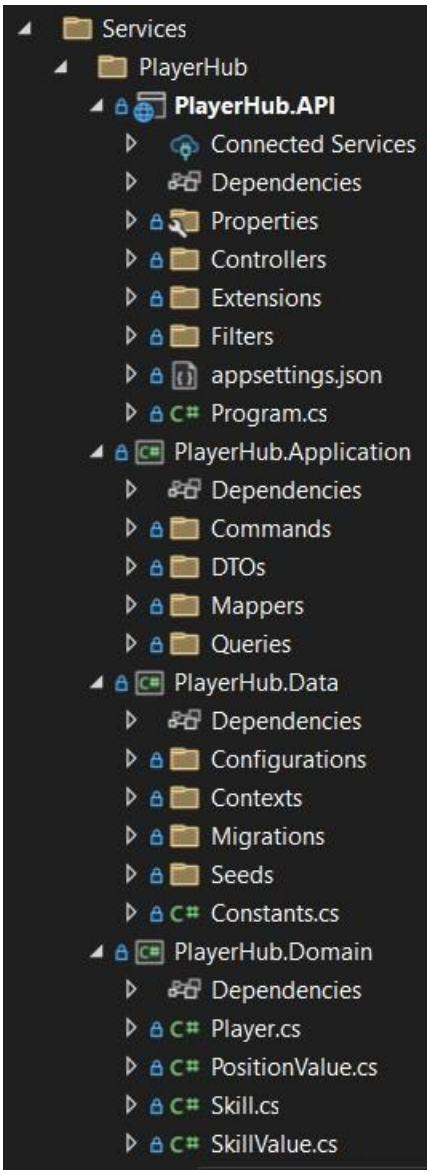
SERVICES: Contains the Application Services (each service is an independent project).

PlayerHub Service:

For a deeper understanding of DDD with CQRS, refer to the following documentation: .NET Microservices. Architecture for Containerized .NET Applications - .NET | Microsoft Learn.

If you have any questions, feel free to ask me.

DESCRIPTION: The project is structured following the principles of Clean Architecture, incorporating a custom Domain Driven Design (DDD) and utilizing Command and Query Responsibility Segregation (CQRS), It is organized into four distinct parts:



DOMAIN LAYER: Following DDD principles, there are four types of Domain Entities: Entity, Aggregate Root, Enumeration, and Value Object, see: Seedwork (reusable base classes and interfaces for your domain model) - .NET | Microsoft Learn.

- **.NET Standard** Library v2.1.0.
- **Domain Entities:** Player (Aggregate Root), Skill (Enumeration).

DATA LAYER:

- **Contexts:** WriteDbContext (read and write), ReadDbContext (read), AppDbContext (read and write over Identity Model).
- **Configurations:** Configure Domain Model relationships within Entity Framework Core to ensure seamless interactions and maintain database integrity.
- **Migrations:** Migrations for Domain Model and Identity Model.
- **Seeds:** Populating a database with an initial set of data.
- Although **UnitOfWork** is not implemented in this layer, its functionality is effectively managed by the data access layer. Coordinates the creation and management of repositories and the data context. It ensures atomic save operations by committing all changes in a single transaction. Additionally, it handles domain events to maintain consistency across aggregates.

APPLICATION LAYER:

- **Commands and CommandHandlers:** Change the state of the application (creating, updating, or deleting data). Use WriteDbContext to save changes into database.
- **Queries and QueryHandlers:** Requests for data and does not alter the application's state. Use ReadDbContext to get data from database.
- **CommandValidators:** Ensure data integrity by enforcing validation rules before processing commands.
- **DTOs:** Transfer data between the client and the server.
- **Mappers:** mapping data between objects (Entity Model to DTO).

API:

- **Controllers:** Handle HTTP requests, manage CRUD operations.
  AccountController: Handles account creation, login, and logout operations, and provides an endpoint for refreshing tokens.
  PlayerController: Manages CRUD operations for Player entity.
- **Filters:** Enforce security policies across the application.
  ApiKeyAuthorizationHandler: Validate the presence and validity of API key in the request.
- **Extensions:**
  ServiceCollectionExtensions: Provides extensions for injecting services as dependencies.
  WebApplicationExtensions: Offers middleware extension methods to enhance web application functionality.

Application Connection String:

Update the connection string credentials, database name, and database source to your SQLServer database:

```
"ConnectionStrings": {
    "AppDbConnection": "Data Source=.;Initial Catalog=PlayerHubDb;Persist Security Info=True;TrustServerCertificate=True;uid=user;pwd=password"
},
```

Application Request Examples:

Note: The x-api-key is mandatory for all client requests.

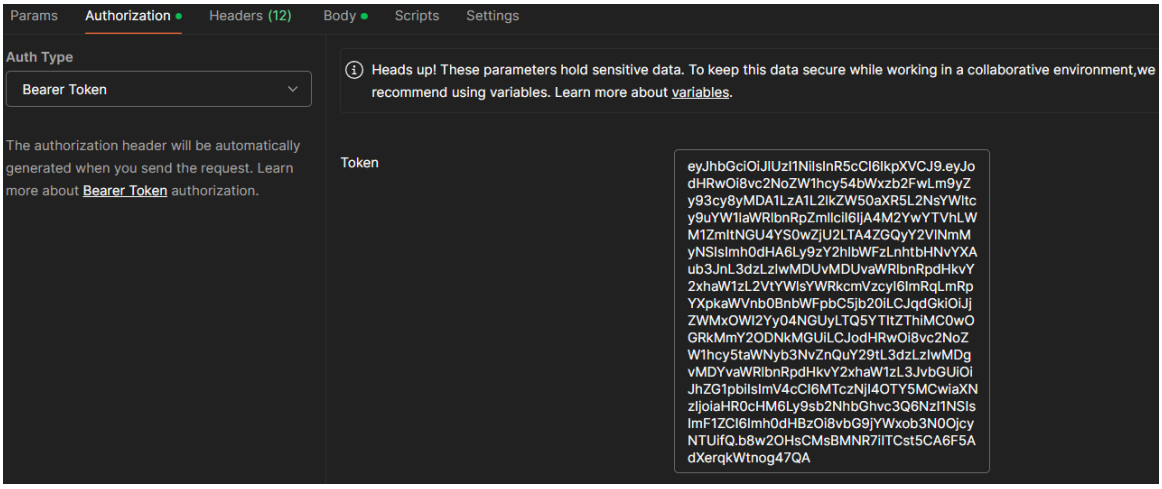
Note: The accessToken is required for all client requests, except for Create User and Login.

Note: Client requests using Postman.

Header:

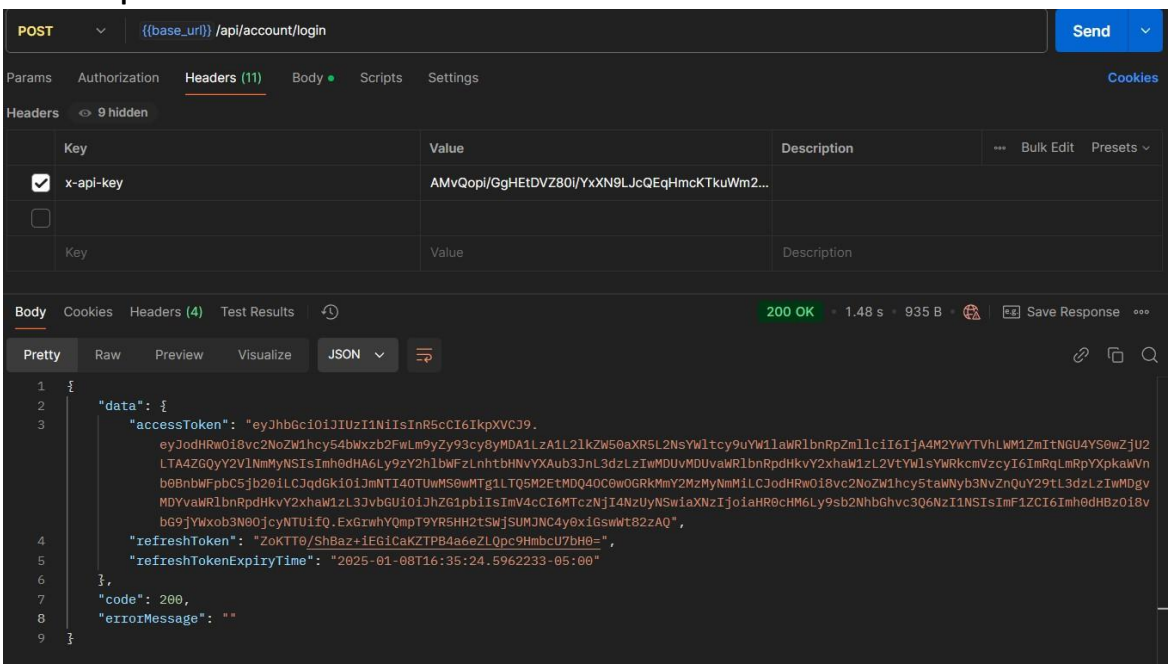
```
x-api-key:AMvQopi/GgHEtDVZ80i/YxXN9LJcQEqHmcKTkuWm2+7VODWkB3JlZsOvpewwWo5Lsw==
```

Access Token: Use the access token as Bearer Token in Postman Authorization.

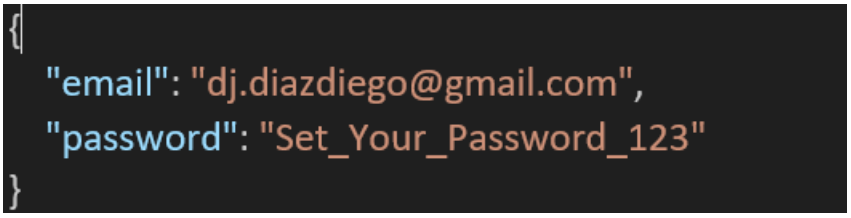


LOGIN:

Client Request:



Body:



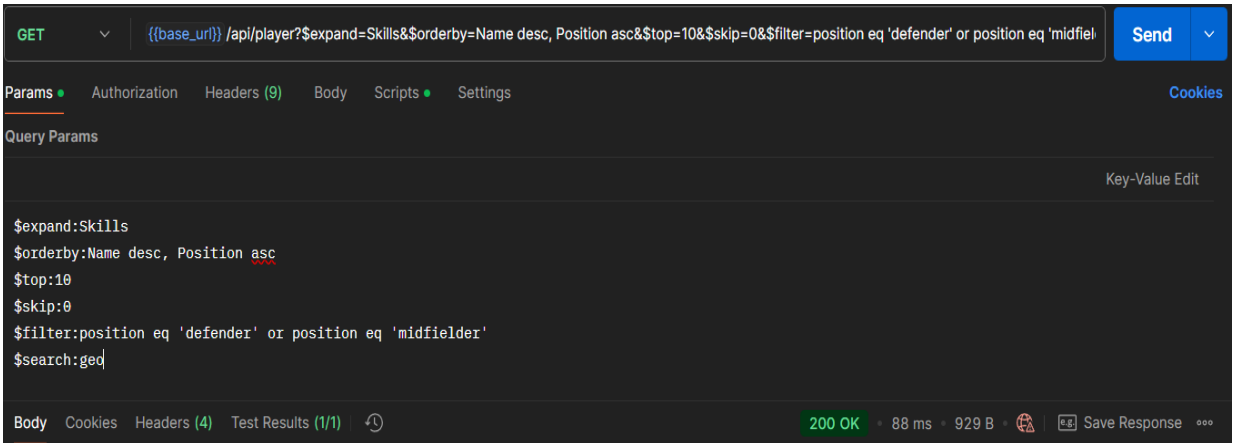
The above credentials belong to a default Admin user created through data seeding.

CREATE A PLAYER:

Client Request:

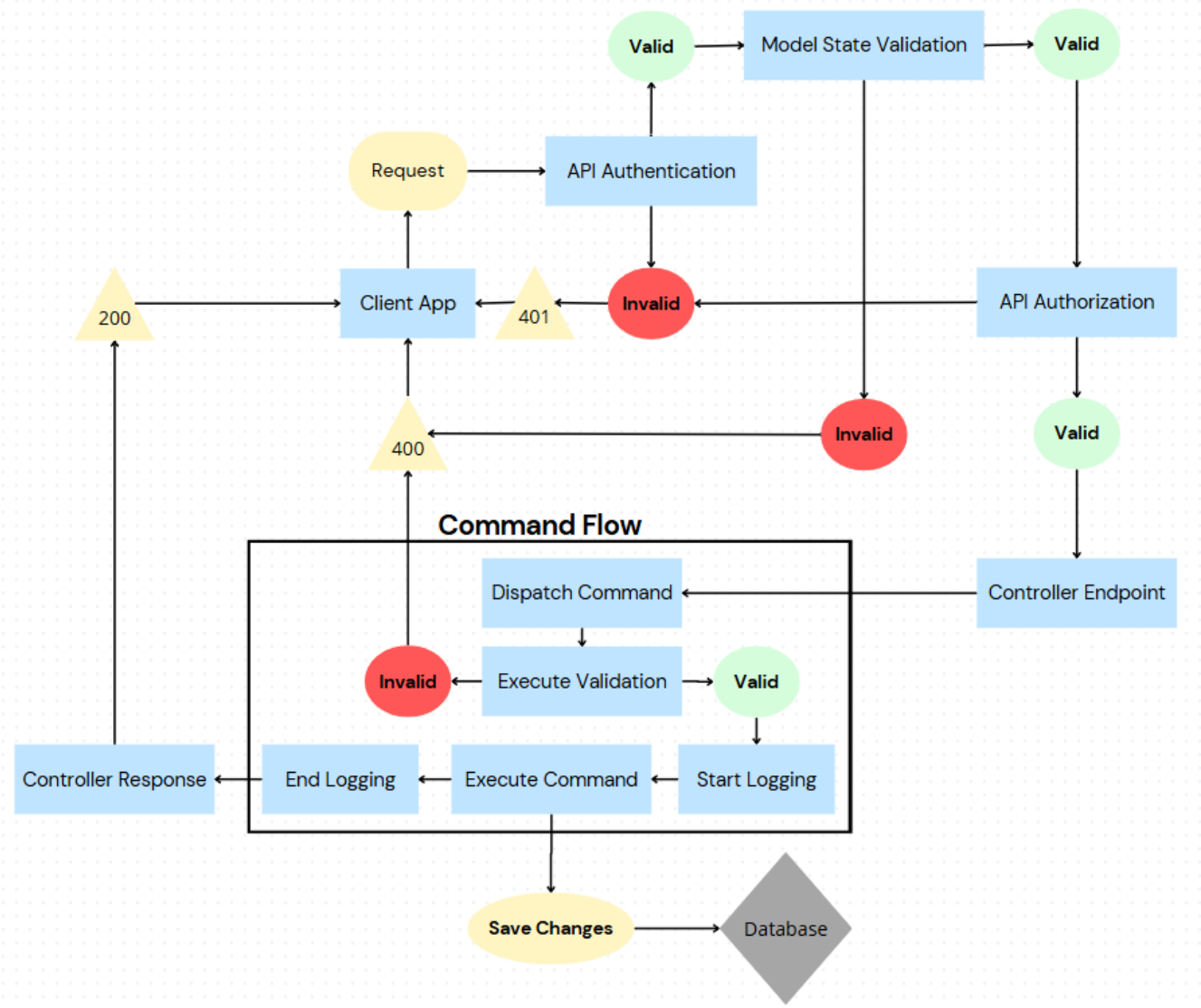
GET PLAYERS:

Client Request:



Note: The endpoint utilizes OData to construct queries. For more information, see: OData overview - OData | Microsoft Learn and Query options overview - OData | Microsoft Learn). For a comprehensive understanding of OData, refer to the following documentation: OData documentation - OData | Microsoft Learn.

Application Basic Command Flow (Start in Client App):


Valid
Model State Validation
Valid
Request
API Authentication
200
Client App
401
Invalid
API Authorization
400
Invalid
Valid
Command Flow
Dispatch Command
Controller Endpoint
Invalid
Execute Validation
Valid
Controller Response
End Logging
Execute Command
Start Logging
Save Changes
Database