**PIDP Documentation**

**OneHealthID**

**Purpose:**

This document serves as a guide for new developer joining the project, providing them with insights into the project technical environment, development workflow and onboarding process. Its purpose is to facilitate a smooth transition for new team members by offering documentation of project process and procedures, enabling them to quickly grasp project requirements, standards and best practices.

**Introduction:**

The OneHealth ID app serves as a crucial tool for doctors, nurses, and various healthcare professionals, streamlining their access to multiple services and logins. Within this application, users encounter different types of logins tailored to the operational structure of the BC government and its associated health authorities.

**Technical Environment:**

* Backend: .NET framework (6.0) & Entity Framework
* Frontend: Nx workspace, Angular framework (v17), Angular material
* Database: Postgres SQL Npgsql Entity Framework Core provider for PostgreSQL(<https://github.com/npgsql/efcore.pg>)
* Containerization: Docker (Docker Desktop is an application installed to provide an Interface to manage the containers)
* Version Control: Git (GUI applications such as GitKraken, GH Desktop leverage)
* IDE-Visual Studio and Visual Studio code
* Database Management Tool- DBWeaver (for managing and interacting with database)
* Hosting: Red Hat OpenShift, deployed using GitHub Actions.

**Codebase Overview:**  
  
Architecture Overview:  
  
Data: This folder has the context class containing all configured tables using the Entity Model. Migration-related instances are stored in the subfolder named Migrations. Model configurations are managed in the configuration subfolder under the data folder.  
  
Extensions: Extension Methods are organized in this folder, categorized based on class types.  
  
Features: In this directory, all API-related code, including controllers, Command, and Query handler classes for each module, are maintained with subfolders for better organization.  
Infrastructure: This folder encompasses commonly used services throughout the project, such as configuring services, injecting interfaces and dependent classes, integrating API calls using HTTP Client, and implementing Authorization services.  
  
Logs: Log files with extensions log & Json are stored in this folder, configured in the program class.  
  
Models: All model classes containing only properties for respective modules/tables are created here. Separate class files are maintained for each module. Table column names are mapped to class properties. Constants are stored in the Lookups subfolder for each model.  
  
Domain Events: Configured under the DomainEvents subfolder for specific classes, these events are utilized in test projects as needed.  
  
Code Organization:  
  
PIDP Project: This project serves as the interface for APIs interacting with the UI Application. All requests and responses occur within this project. Validation and authorization are performed using Keycloak, with token generation facilitated through JWT.  
  
Design Pattern: Following the CQRS pattern (Command and Query Responsibility Segregation)

Naming convention, file organization, and Comments:

* Use camel case while declaring method parameters, properties creation and defining model.
* Static files or logged files should be maintained in separate folders.
* Only include comments when necessary to provide clarification or context for complex logic, prioritize self-explanatory code when possible.

**Development Workflow:**

Ticket Management:

* Agile methodology is followed, with the team working during 2-week sprint.
* Sprint planning sessions determine the sprint size, included tickets and priority items.
* Prior to sprint planning, tickets are discussed, refined, and scoring during a refinement session.
* After a ticket is assigned in JIRA, developers review the ticket details, including requirements, acceptance criteria, and any relevant documentation.
* Once the scope and understanding are clear, developers proceed to work on the ticket.

Version Control

* Developers utilize Git for version control to manage changes to the codebase.
* Before starting work on a ticket, developers create a new branch from the develop branch, usually following a naming convention that includes the ticket number and a brief description. (Use all lower case for your branch names to avoid casing issues)
* Throughout the development process, developers commit their changes to the local branch, providing meaningful commit messages to track progress and changes.
* Periodically, developers push their local branches to the remote repository (e.g., GitHub) to collaborate with other team members and ensure backup of their work.
* Once the ticket is completed and ready for review, developers create a Pull Request (PR) to merge their changes into the develop branch, triggering the code review process.

Code Review Process:

* Once the developer considers the task completed, they create a pull Request (PR) to merge their changes into the develop branch.
* Other developers are assigned to review the PR.
* Reviewers provide feedback through comments or during a call.
* After addressing any feedback and ensuring the code meets the projects standards, the PR is approved.
* Upon approval, the changes are merged into the develop branch.

Deployment Process:

* After merging into develop branch, changes are deployed to the test environment for further testing.
* Upon successful testing, changes are deployed to the production environment.

**Setup Instructions:**

[moh-pidp/workspace/README.md at develop · bcgov/moh-pidp (github.com)](https://github.com/bcgov/moh-pidp/blob/develop/workspace/README.md)

[moh-pidp/backend/webapi/README.md at develop · bcgov/moh-pidp (github.com)](https://github.com/bcgov/moh-pidp/blob/develop/backend/webapi/README.md)

**My Onboarding Process:**

* Upon joining the project, I received guidance from the development team to set up my local development environment.
* The Scrum Master provided a walkthrough of the project.
* I was introduced to the project's version control system and given access to the code repository for collaboration.
* As part of the onboarding process, I familiarized myself with the project's Agile methodology, sprint planning, and iterative development practices.
* The development team has provided valuable feedbacks and thorough reviews on Pull requests, enhancing my understanding of the codebase and refining my skills.
* I participated in team meetings, stand-ups, and retrospective sessions to gain insights into project progress and contribute to continuous improvement efforts.