NTT DATA

.Net

Plano de formação - AIS

2022-07-08

FUTURE
AT HEART

Plano formação – .Net

Âmbito

- Este documento contém informação sobre o plano de formação .Net Core para os consultores AIS

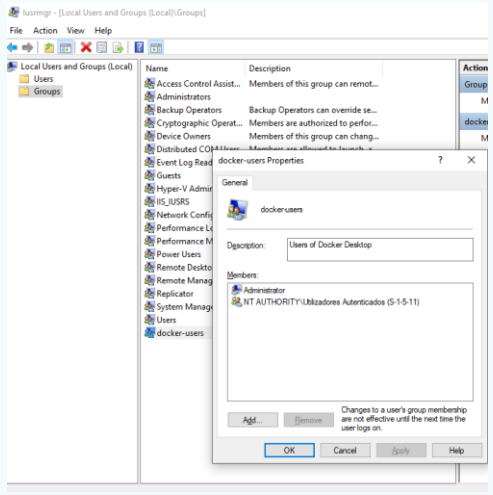
- As formações indicadas são da Udemy, sendo que adicionalmente os consultores podem alavancar os seus conhecimentos com outras fontes de informação
    - Udemy é uma plataforma de e-learning que permite escolher e realizar um conjunto de formações, nas mais diversas temáticas, possibilitando a criação de um learning path personalizado
        - https://business.udemy.com/
        - Para usufruíres desta licença, solicita aqui

- Algum dos pré-requisitos de software a instalar requere permissões de administração
    - **Privilégios de Administrador**: Por omissão o teu utilizador não tem privilégios de administração sobre o computador. Caso seja necessário, deves abrir uma incidência no click na categoria Request administrator local password.

Requisitos

Programas & Ferramentas relevantes

- Visual Studio Community 2022
- .Net Core 6.0
  - pode ser instalado via Visual Studio
- Postman
  - Utilitario para testar/invocar REST API
- Docker Desktop on Windows
  - Criação e gestão de containers
  - Linux containers:
    - Manual installation steps for older versions of WSL
    - Seguir as instruções WSL, step 4 e 5 (6 é opcional)



Adicionar o grupo do domínio local “Utilizadores Autenticados (Authentication Users)” ao grupo “ “docker-users”

- Lançar “command prompt” como “administrador”
- Dar o comando: lusrmgr
- Encontrar o grupo “Authentication Users” e adiciona-lo
- Pudera ser necessário fazer reboot

- DBeaver
  - IDE para base dados
- GitHub Desktop
- Offset explorer

Opcionais

- Node.js
- Visual Code
- OpenSSL
  - Biblioteca que implementa as funcionalidades genéricas de criptografia
  - Possibilta gerar certificados SLL
- Chcolatey
  - software management automation
  - A partir deste utilitário pode-se instalar outras ferramentas/packages como “GNU make” (makefile)

Nota: as formações podem obrigar a instalação de outro software não indicado na lista acima

Suporte

Recursos

- Try .NET Online
  - Runnable .NET code on your site
- C# documentation
  - Learn how to write any application using the C# programming language on the .NET platform.
- .NET documentation
  - Learn about .NET, an open-source developer platform for building many different types of applications.
- Learn .NET | Free tutorials, videos, courses, and more
  - Free tutorials, videos, courses, and more for beginner through advanced .NET developers.
- C# Cheat Sheet
  - C# cheat sheet includes symbol syntax and methods to help you using C#


NTT DATA
Formação - .Net Backend

Cursos

Cursos formação .Net/C#

- C# Basics for Beginners: Learn C# Fundamentals by Coding
  - Learn the fundamentals of C# and .NET Framework
  - Work with primitive types and expressions
  - Work with non-primitive types (classes, structs, arrays and enums)
  - Learn the difference between value types and reference types
  - Control the flow of programs using conditional statements
  - Use arrays and lists
  - Work with files and directories
  - Work with text
  - Work with date and time
  - Debug C# applications effectively

Efetuar formação ou rever matéria que acharem necessário

- C# Intermediate Programming: Classes, Interfaces and OOP
  - Work with classes, constructors, fields, properties, methods and indexers
  - Use encapsulation to improve the robustness of the code and reduce the impact of change
  - Re-use code using inheritance and composition
  - Understand the problems with inheritance and how composition solves these problems
  - Change the behaviour of an application by extending its code, rather than changing it
  - Develop loosely-coupled, testable and extensible applications using interfaces

- C# Advanced Topics: Prepare for Technical Interviews
  - Understand advanced C# features and apply them at work
  - Master the confusing C# constructs: Events, Delegates, Lambda Expressions, LINQ, Async/Await and more!

Cursos

Cursos auxiliares

- The Complete SQL Bootcamp 2022: Go from Zero to Hero
  - Use SQL to query a database
  - Use SQL to perform data analysis
  - Be comfortable putting SQL and PostgreSQL on their resume
  - Learn to perform GROUP BY statements
  - Replicate real-world situations and query reports

Efetuar formação ou rever matéria que acharem necessário

- Learn Git by Doing: A step-by-step guide to version control
  - Track and Modify projects using Git
  - Revert/Reset their project to a previous version
  - Create multiple versions of a project and merge them together
  - Collaborate and share projects using Github
  - Understand when and why to use Git and/or Github for version control on a project
  - Recognize when to use what Git command in the terminal
  - Use advanced git commands for more complex tracking and editing scenarios

- Docker for the Absolute Beginner - Hands On - DevOps
  - Beginner level introduction to Docker
  - Basic Docker Commands with Hands-On Exercises
  - Build Docker images using Dockerfiles with Hands-On Exercises
  - Understand what Docker Compose is
  - Build Application stack using Docker Compose Files with Hands-On Exercises
  - Understand what Docker Swarm is

- Unit Testing for C# Developers: Unit Testing with Nunit and Moq
  - Learn unit testing from scratch
  - Tips and tricks to write clean, maintainable and trustworthy tests
  - Write loosely-coupled and testable code
  - Refactor legacy code towards testable code
  - Understand and implement dependency injection
  - Use mocks to isolate code from external dependencies
  - Apply the unit testing best practices
  - Learn the anti-patterns to avoid

Efetuar formação ou rever matéria que acharem necessário

Cursos

Cursos .Net/REST API

- Software Architecture: REST API Design - The Complete Guide

  - Best Practices of professional REST API
  - Correct use of HTTP Verbs, URL structure and response codes
  - Using world-leading documentation engine to document your REST API
  - Integrating Authentication & Authorization capabilities into REST API
  - Performance techniques to speed up the response from the REST API
  - Using Postman to test your REST API

- Build ASP.NET Core Web API - Scratch To Finish (.NET 6)

  - Learn, Understand and Create ASPNET Core Web API From Scratch
  - Building scalable REST Web APIs from scratch using ASPNET CORE and C#
  - Learn and Apply Entity Framework Core to perform CRUD operations on a SQL Server database
  - Use Entity Framework Core in a code first approach
  - Understand and Apply the Repository Pattern
  - Use Domain Driven Design (DDD) approach to create domain first models and project
  - Understand RESTful Principles and Apply them in ASPNET Core Web API
  - Understand Best practices and Clean Coding Techniques, Know Shortcuts and Tips and Tricks
  - Add Validations In ASPNET CORE RETS API
  - Use popular third-party libraries such as AUTOMAPPER and FLUENT VALIDATION
  - Understand and Use Interfaces, Inheritance, Dependency Injection etc
  - Understand and Implement Authentication and Role based Authorization to Authenticate and Authorize the ASPNET Core Database
  - Create JWT tokens to Authenticate API
  - Test ASPNET Core Web API using Swagger and Postman

Projecto – Open Bank API

Âmbito

Criação de uma Web API, “Open Bank API”, para fornecimento de funcionalidades de gestão de clientes, contas bancarias e transações

Funcionalidades

- Login
  - Autenticação do user
- Create User
  - Cria user para acesso API
- Create Account
  - Cria conta para o utilizador
- List Accounts
  - Obter lista de contas associado ao utilizador
- Get Account By Id
  - Retorna conta (+movimentos) do identificador passado associado ao utilizador
- Transfer
  - Transfere montante da conta de origem para conta destino

Requisitos funcionais

Regras gerais

- Acesso a informação requere previa autenticação
- Deve ser validado se o cliente está associado a conta para o qual está a ser pedido execução da operação (seja consulta conta, transferência, …)
- Informação sigilosa deve ser armazenada de forma segura

Regras por funcionalidade

- Create User
  - Objectivo:
    - Criação credenciais (username/password) do novo utilizador para acesso API
  - Regras:
    - Deve validar se o “username” já existe
    - Password deve ser guardada na base dados encriptada (hash)
- Login
  - Objectivo:
    - Autenticação do user via credenciais (username/password)

Requisitos funcionais

Regras por funcionalidade (cont.)

- Create Account
  - Objectivo:
    - Cria conta para utilizador autenticado

- List Accounts
  - Objectivo:
    - Retorna lista das contas que existam associadas ao utilizador autenticado

- Get Account By Id
  - Objectivo:
    - Retorna detalhe (movimentos) da conta do identificador (id) indicado caso esteja associado ao utilizador autenticado
  - Regras
    - Validar se o id da conta está associado a conta do utilizador autenticado

Requisitos funcionais

Regras por funcionalidade (cont.)

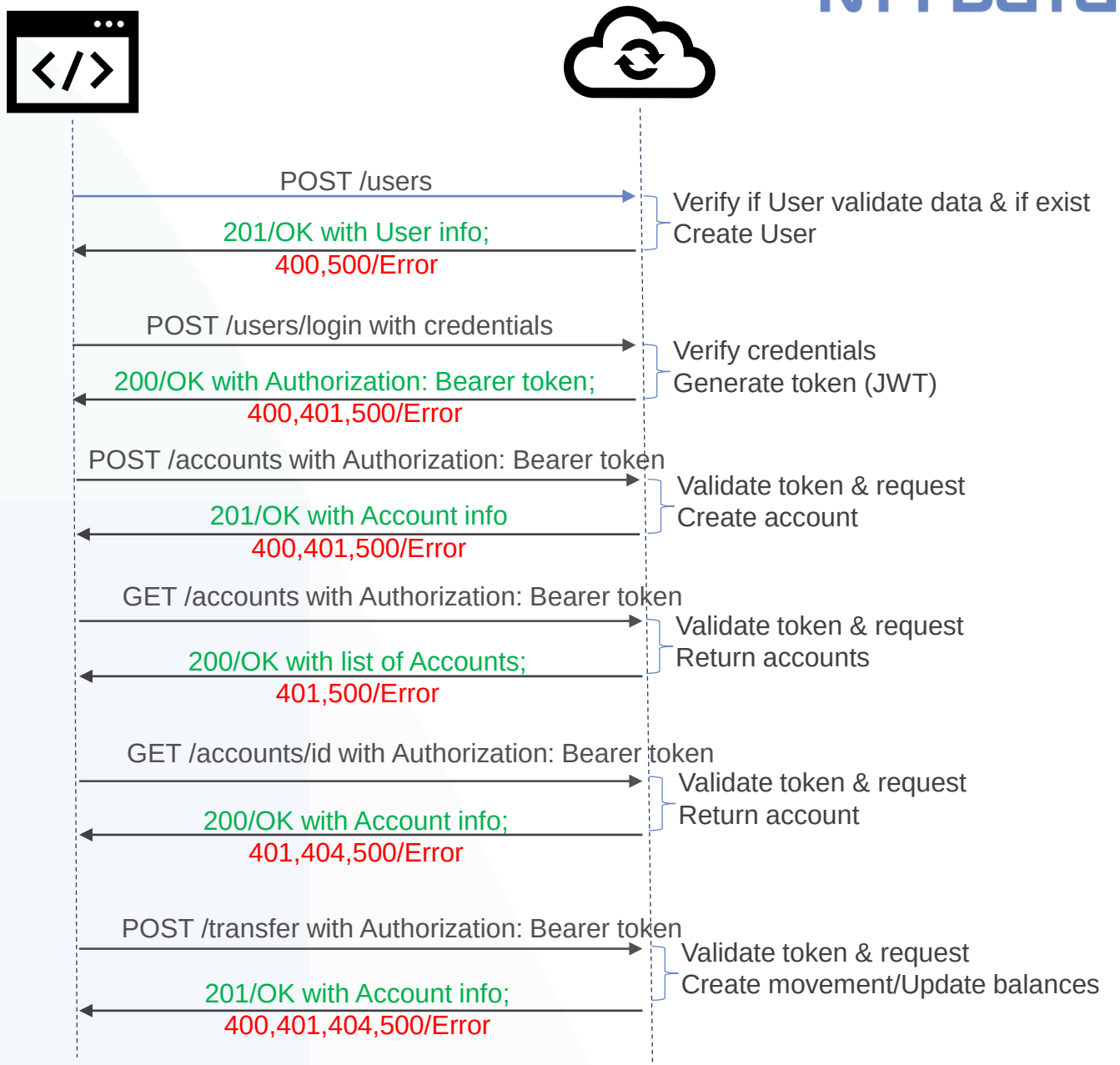
- Transfer
  - Objectivo:
    - Transfere montante da conta de origem para conta destino
  - Regras:
    - Deve validar se utilizador autenticado está associado a Conta origem
    - Deve validar se a moeda da conta de origem é igual a de destino
    - Deve validar se existe montante disponível na conta origem
    - Deve criar movimento debito na conta origem e de credito na conta destino
    - Deve incrementar o saldo da conta destino e decrementar o saldo conta origem

Requisitos técnicos

- No diagrama indica-se genericamente os fluxos dos das funcionalidades requeridas
- No Swagger file em anexo encontra-se a especificação técnica da API

HTTP Response codes

200 – OK (GET/DELETE)
201 – Created (POST)
400 – Bad request (bad payload/missing parameters)
401 - Authentication required
402 – Forbidden (no permissions)
404 – Not found (action don’t exist)



Requisitos técnicos

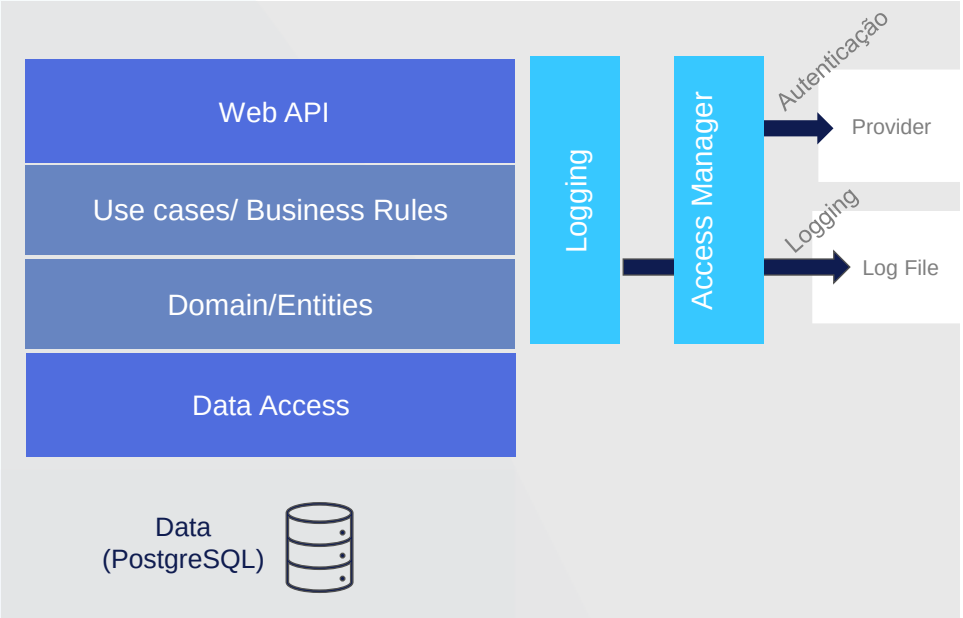
- Database: PostgreSQL
  - A informação negocio deve ser preservada em base dados

- Web API: deve ser desenvolvida em REST com recurso .Net Core/C#
  - Deve ser usado JWT (Json Web Token) para validar o *requester* (quem invoca API)

- Será validado a arquitetura da solução aplicacional,
  - A solução deve estar modulada em componentes independentes e intercambiáveis, de modo que cada um contenha apenas o que é necessário para executar o âmbito da sua funcionalidade
  - A solução deve ter em conta na modulação, o principio de “responsabilidade única”
  - Padrões de desenho, bem como componentes (Libraries/Nuget Packages) a usar ficam aos critério do implementador (ex: ORM para gestão com base dados)

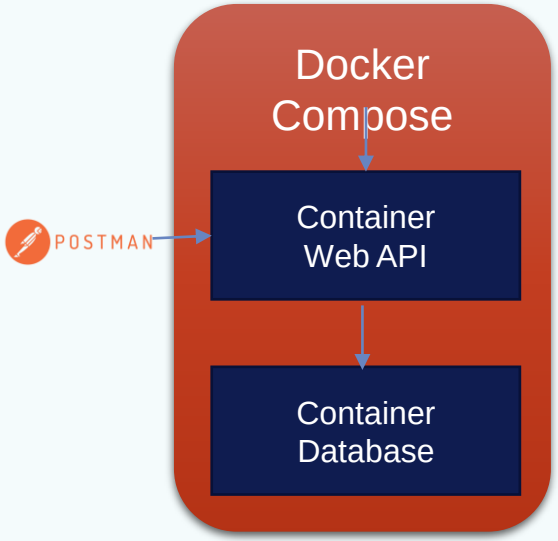
Requisitos técnicos

- A solução apresentada, em termos arquitetura logica, deve estar devida em 2 componentes principais:
  - Web API
  - Database



Stack tecnológico

***Web API*:**
ASP.Net Core v6.0
RESTful Web Services/JSON
***Business/Data Access*:**
.Net Core v6.0
Base de dados:
PostgreSQL v14



- A solução final deve estar “contentorizado”, com recurso Docker, de acordo arquitetura logica (ver diagrama)
  - Containers serão Linux

Requisitos técnicos

- Devem ser registados todos os eventos/ações que permitam detetar anomalias/exceções de modo a possibilitar a sua analise
  - O log pode ser em ficheiro
  - A informação a registar, para alem anomalias/exceções deve conter informação necessária de contexto mas sem informação que seja considerada sigilosa (de acordo a lei RGPD)

- O código deve estar num sistema de controle de versões
    - Criar um repositório no GitHub publico

Formação – Architecture & Best Pratices

Cursos

Arquitetura e Desenho soluções

- ASP.NET Core - SOLID and Clean Architecture
  - Implement SOLID Principles
  - ASP .NET Core MVC and API design
  - Advanced Tools - MediatR, Automapper, Fluent API and Validation
  - Custom Exceptions and Global Error Handling
  - Custom .NET Core Middleware
  - Use Swagger for API Documentation
  - Implement CQRS Pattern
  - Use Identity and JWT To Secure Application API
  - Build API Client Secure Application
  - Moq and Shouldly Frameworks
  - Unit Testing

- Master the Art of Writing Clean Code in C#
  - Give best names for functions, variables and other API members
  - Understand and rely on programming metaprinciples such as DRY, YAGNI, KISS and others
  - Write clean functions
  - Detect common architectural smells and refactor the problems
  - Apply principles of functional programming
  - Apply Dependency Injection and avoid common DI-related smells
  - Write clean unit tests
  - Practice Test-Driven Development

Cursos

Cursos auxiliares

- Azure DevOps Fundamentals for Beginners
  - Create an Azure DevOps organization
  - Align Azure DevOps work items using Agile, Scrum, or Basic work processes
  - Integrate an Azure DevOps code repository with GitHub
  - Fork and clone code using multiple tools
  - Understand the basic vocabulary of DevOps: what it is and why it matters
  - CI/CD: Understand how Pipelines facilitate Continuous Implementation and Continuous Deployment
  - Commit code changes and track Pull Requests
  - Push a code Repo from the command line of an Integrated Development Environment (IDE)

Projecto – Desenvolvimentos adicionais

Requisitos Funcionais

- Token–refresh
  - Objectivo:
    - Implementar funcionalidade que permita “revalidar” a permissão de acesso (prolongar o tempo de expiração do token) sem haver necessidade de o utilizador ter que efetuar novo login
  - Regra:
    - Implica extender “login” para passar retornar “token refresh”

- Submit document
  - Objectivo:
    - Implementar funcionalidade que permita adicionar 1 documento digital associado a conta (Account) do user
  - Regras:
    - Deve ser validado se o cliente está associado a conta para o qual está a ser pedido execução da operação
    - Deve aceitar documentos do tipo PDF e PNG, tamanho máximo 2MB

Requisitos Funcionais

- Get documents
  - Objectivo:
    - Implementar funcionalidade que permita obter a lista documentos associado a conta (Account) do user
  - Regras:
    - Deve retornar o nome documentos, tipo e data submissão e id

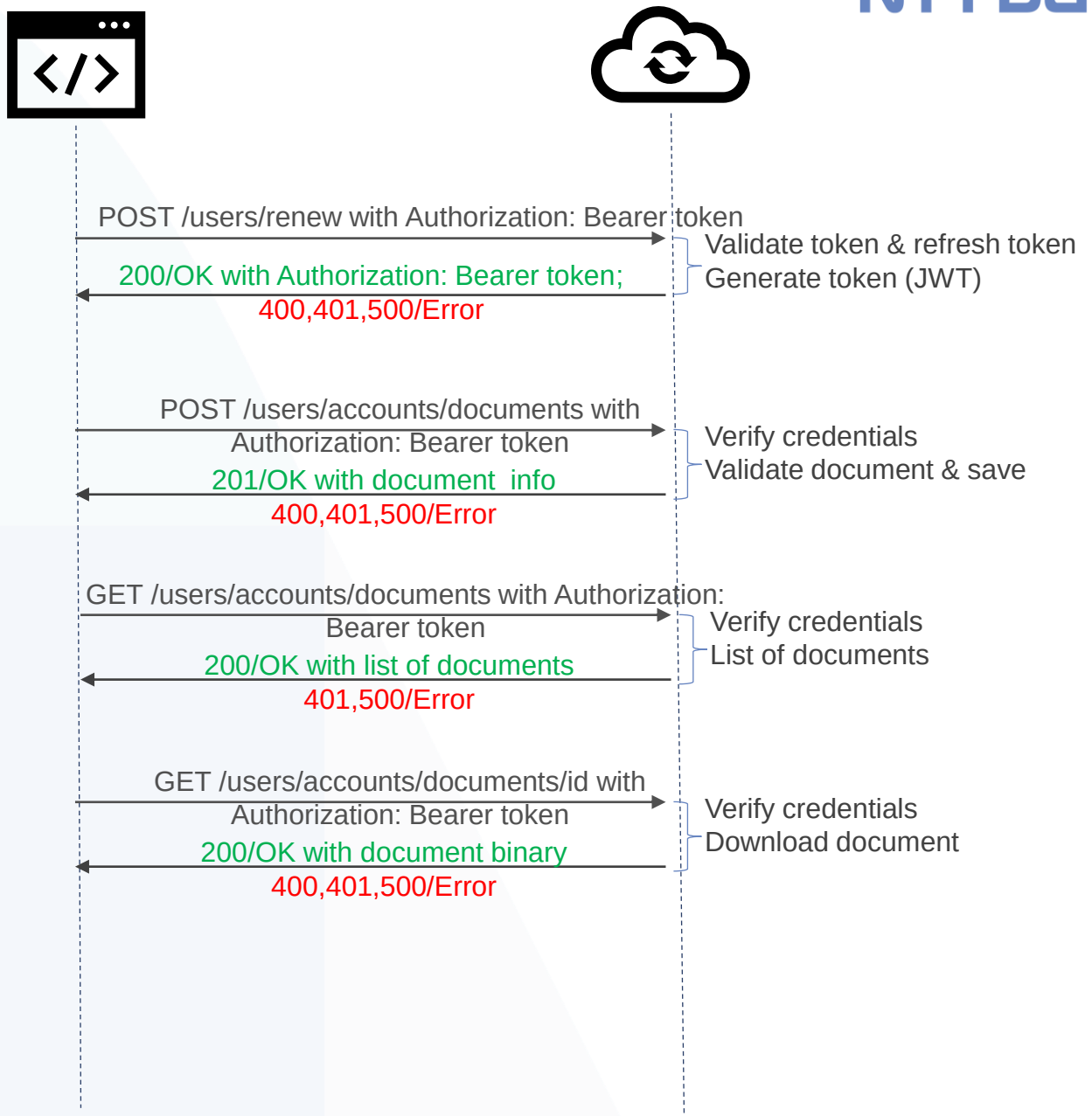
- Download document
  - Objectivo
    - Implementar funcionalidade que permita fazer download de um documento associado a conta (Account) do utilizador
  - Regras
    - Validar se o id do documento está associado a conta do utilizador autenticado

Requisitos técnicos

- No diagrama indica-se genericamente os fluxos dos das funcionalidades requeridas

HTTP Response codes

200 – OK (GET/DELETE)
201 – Created (POST)
400 – Bad request (bad payload/missing parameters)
401 - Authentication required
402 – Forbidden (no permissions)
404 – Not found (action don’t exist)



Requisitos técnicos

- Unity test
  - Implementar testes unitários para validar os componentes
  - Framework a usar fica ao critério do implementador
  - Criar mock para “virtualizar” repositório
  - Deve criado cenários para testar as varias regras negocio dos componentes Application/UseCases
  - Casos de testes:
    - Accounts
      - Get All Accounts
        - OK
        - Exception (internal server error)
      - Get Account By Id
        - OK
        - Conta inexistente (not found)
        - Exception (internal server error)
      - Create Account
        - OK
        - Exception (internal server error)

Requisitos técnicos

- Unity test
    - Transfer
        - OK
        - Utilizador não está associado a Conta origem (bad request)
        - Moeda da conta de origem é diferente a de destino (bad request)
        - Montante disponível na conta origem inferior ao montante transferência (bad request)
        - Exception (internal server error)


NTT DATA
Formação - Design patterns

Cursos

Arquitetura e Desenho soluções

- Design Patterns in C# and .NET: Learn Solutions to Common Problems
  - Recognize and apply design patterns
  - Refactor existing designs to use design patterns
  - Reason about applicability and usability of design patterns

Recursos auxiliares

The Catalog of Design Patterns


NTT DATA
Formação - Kafka

Cursos

Apache Kafka

- Apache Kafka Fundamentals
  - Introduces the basic Apache Kafka elements and APIs
  - Kafka ecosystem.
  - Topics, partitions, brokers, replication
  - Producers & consumers
- Microservices using Kafka
  - How to run Kafka on your windows 10
  - C# Implementation using Confluent.Kafka
  - *Playlist com 3 videos*

Recursos auxiliares

- Using Apache Kafka with .Net
- Working with Apache Kafka in ASP.NET 6 Core
- Apache Kafka and .NET - Getting Started Tutorial (confluent.io)

Projecto – Kafka

Âmbito

- Criação de um componente responsável pelo envio notificações de confirmação do resultado das operações (Transferência)

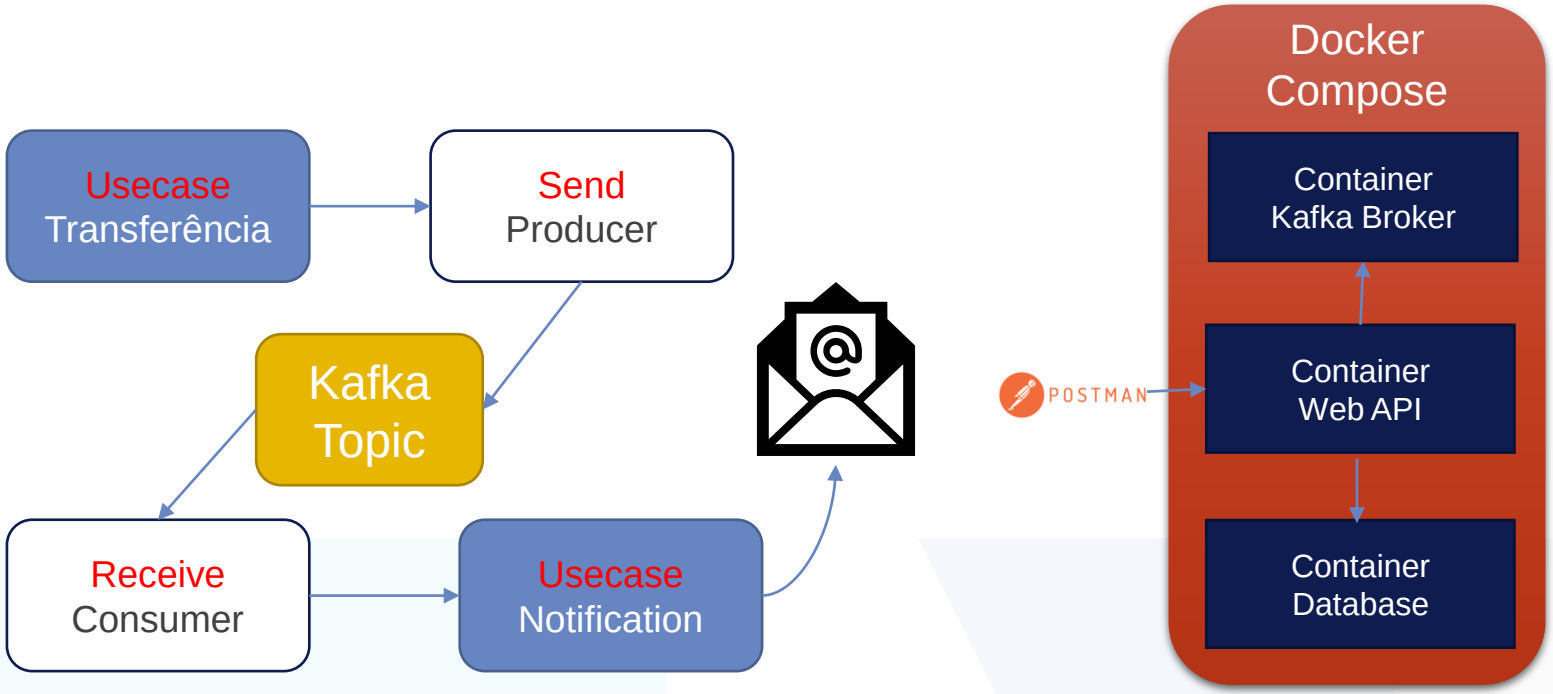
- **Requisitos funcionais**
  - Quando da realização de uma operação (transferência), deve ser enviado uma notificação , via email, para os clientes (users) associados a conta de destino e origem da operação
  - Regras
    - Se o cliente (user) for o mesmo da cota origem e destino, deve ser envaido 1 notificação com a informação da operação
    - Caso contrario deve ser enviado 2 notificações com informação respetiva (debito ou credito), 1 para cada cliente (user) associado

Âmbito

- **Requisitos técnicos**
  - Para suporte a criação e gestão de eventos de notificação, será usado o sistema Kafka
  - Producer: componente responsável pela “produção” de evento por cada notificação, o qual será inserido no tópico Kafka criado para o efeito
  - Consumer: componente responsável por “consumir” os eventos do topico Kafka e por cada evento enviar uma notificação por email

Stack tecnológico

***Web API*:**
ASP.Net Core v6.0
RESTful Web Services/JSON
***Business/Data Access*:**
.Net Core v6.0
Data stream:
Kafka



- A solução final deve estar “contentorizado”, com recurso Docker, de acordo arquitetura logica (ver diagrama)
  - Containers serão Linux


NTT DATA
Formação - .Net Frontend

Cursos

Cursos auxiliares

- JavaScript Basics for Beginners
  - Understand the fundamental concepts in JavaScript
  - Learn problem-solving skills
  - Learn and apply the best practices
  - Avoid common pitfalls and mistakes other JavaScript developers make
  - Write solid JavaScript code

Cursos .Net/MVC

ASP.NET Core 6.0 Course - MVC - Blazor - Razor - EF Core

- Development of complete Web Applications using ASP NET Core
- Backend Development using ASP NET Core 6 and Entity Framework
- MVC Pattern using Razor Pages and Blazor
- RESTful API Development
- Roles and Accounts
- Deployment of Web Applications

Projecto – Open Bank Web

Âmbito

- Criação de uma Web App, “Open Bank Web”, para fornecimento de funcionalidades de gestão de clientes, contas bancarias e transações via web, tendo em conta as funcionalidades previstas são as servidas pela Web API “Open Bank API”

- Aplicação é composta por várias funcionalidades, algumas acedidas após o utilizador realizar o login (área privada) outras não (área publica)

- Área Publica
    - Login
    - Create User
- Área Privada
    - Disponibilizadas através do Menu Principal, composto por 2 opções:
        - Contas (Accounts: para aceder as contas do utilizador)
            - Create Account
            - List Accounts
            - Get Account Details
        - Transferências (Transfer: para aceder as funcionalidades de transferências bancarias)
            - Transfer

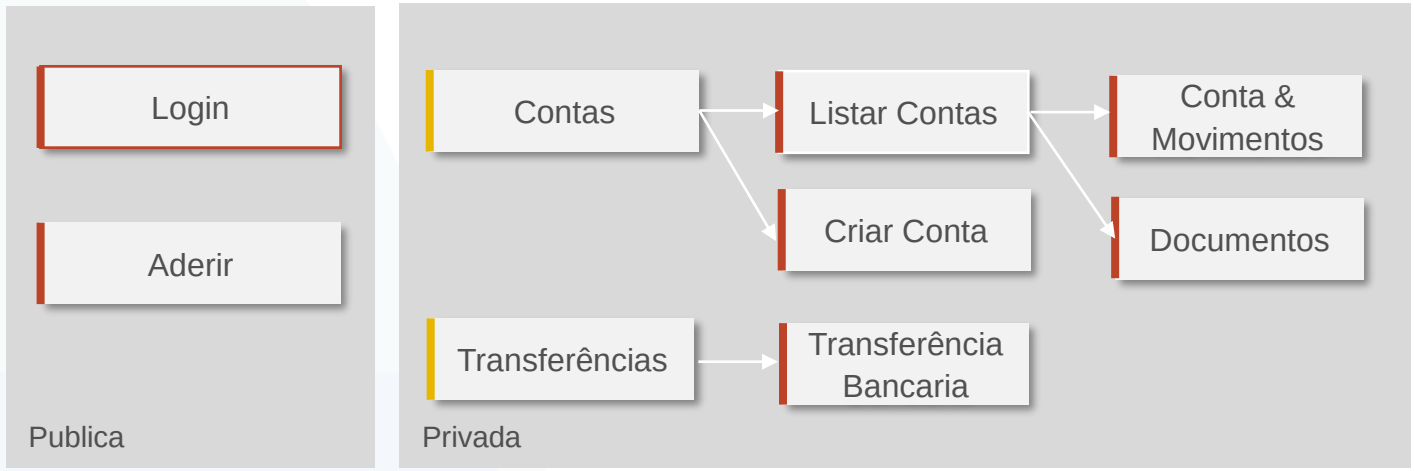
Requisitos funcionais

Regras gerais

- Acesso a informação requere previa autenticação
- Implementar solução para suporte multilingue
    - Configurar para demostração nas línguas PT e EN

Regras por funcionalidade

- Aderir (área publica)
    - Para criação de user para acesso a aplicação
- Login (área publica)
    - Autenticação do user via credenciais (username/password)

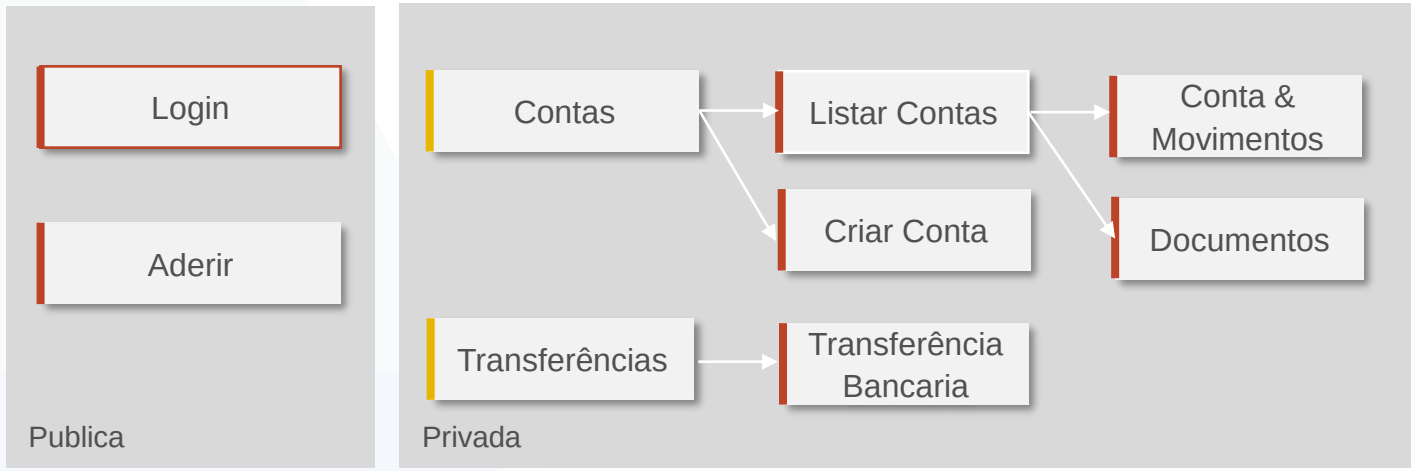


Legenda: Menu Ecrã

Requisitos funcionais

Regras por funcionalidade (cont.)

- Criar Conta
  - Cria conta para utilizador autenticado
- Listar Contas
  - Necessita de estar autenticado
  - Retorna lista das contas que existam associadas ao utilizador autenticado
- Conta & Movimentos
  - Necessita de estar autenticado
  - Retorna dados detalhe da conta selecionada
- Documentos
  - Necessita de estar autenticado
  - Lista documentos associado a conta
  - Permite fazer visualizar e download documento



Legenda: Menu Ecrã

- Transferência bancaria
  Necessita de estar autenticado
  Transfere montante entre 2 contas que tenham a mesma moeda

Requisitos técnicos

- Incorporar os requisitos técnicos da Web API “Open Bank API”

- Web App: fica ao critério do implementador a seleção da base tecnológica a usar para desenvolvimento Web App que servira como demostrador da Web API
  - Exemplo: Blazor Server com razor pages
  - Exemplo2: Node.js com recurso React/Vue.js/Angular.js

- Será validado a arquitetura da solução aplicacional,
  - A solução deve estar modulada em componentes independentes e intercambiáveis, de modo que cada um contenha apenas o que é necessário para executar o âmbito da sua funcionalidade
  - A solução deve ter em conta na modulação, o principio de “responsabilidade única”
  - Padrões de desenho, bem como componentes (Libraries/Nuget Packages) a usar ficam aos critério do implementador (ex: ORM para gestão com base dados)

Recursos

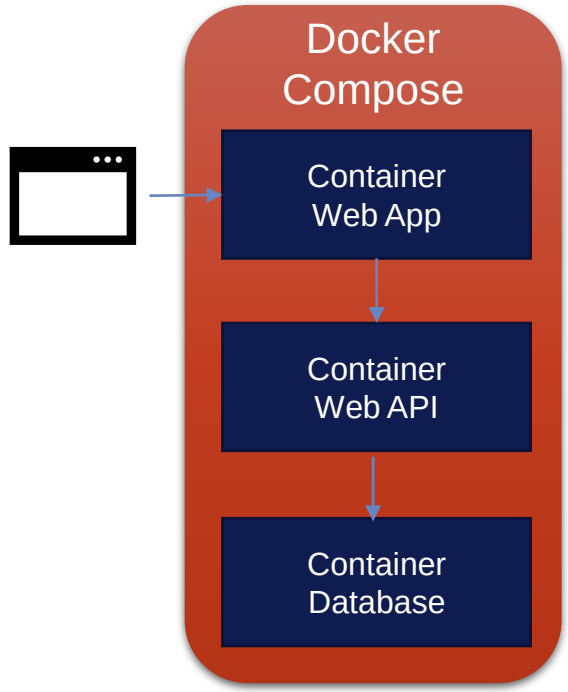
- ASP.NET Core Blazor globalization and localization

Requisitos técnicos

- Devem ser registados todos os eventos/ações que permitam detetar anomalias/exceções de modo a possibilitar a sua analise
  - O log pode ser em ficheiro
  - A informação a registar, para alem anomalias/exceções deve conter informação necessária de contexto mas sem informação que seja considerada sigilosa (de acordo a lei RGPD)

- O código deve estar num sistema de controle de versões
  - Criar um repositório no GitHub publico

- A solução final deve estar “contentorizado”, com recurso Docker, de acordo arquitetura logica
  - Container será em Linux



Formação - CI/CD DevOps

Cursos

DevOps & CI/CD

- Introduction to Continuous Integration & Continuous Delivery
  - Understand what Continuous Integration and Continuous Delivery is
  - Obtain a firm understanding of DevOps
  - Understand the competitive advantages of using CI and CD in your organization
  - Find out why companies like NetFlix and Amazon use CI and CD in their daily workflow
  - Find out about the popular tools for integrating CI and CD into your workflow
  - See an actual case study of a real project using CI/CD from start to finish

- Learn Azure DevOps CI/CD pipelines

  - What is DevOps and CI/CD
  - Creating CI/CD pipelines for Web, Mobile and Container apps
  - Including the Database into the pipelines
  - Configure deployment to Azure Cloud
  - Run unit, functional and load tests as part of the pipeline
  - Analyse source code using Sonar
  - Use Infrastructure as Code (IaC) from CI/CD with ARM templates
  - Use Configuration as Code (CaC) in Azure DevOps
  - Create Dev-Test-Prod environments
  - Secure the pipelines sensitive data
  - Use Infrastructure as Code (IaC) from CI/CD with Terraform

Projecto – CI/CD

Âmbito

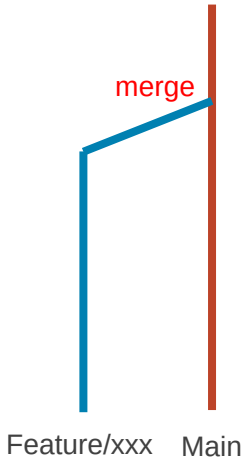
- Criação de processo que permitam dar suporte CI/CD (Continuous Integration / Continuous Delivery) via Azure DevOps pipelines

Requisitos tecnicos

- Criar 1 branch “protegida” no repositório
  - “main”
  - *As restantes branch que sejam criada são temporárias, associadas a desenvolvimentos de funcionalidades/requisitos (“feature”).*

- Pipelines
  - Pipelile “feature” - ao fazer push para o repositório o pipeline deve executar os steps:
    - Unit tests (opcional)
    - Build App

  - Pipeline “main” - ao fazer push para o repositório o pipeline deve executar os steps
    - Build App
    - Build Docker
    - Migrate DB
    - Deploy



GitHub Actions

Azure DevOps & Pipelines

Azure KeyVault (secret manager)

Docker

NTT DATA

Obrigado