

**Trabalho Prático II – G09**

**Integração de Sistemas de Informação**

**Alunos:**

**Allan Sales Aleluia, a21990**

**Francisco Moreira Rebêlo, a16443**

**José Carlos Paschoal, a15926**

**Docente:**

**Óscar Ribeiro**

|  |  |
| --- | --- |
| Versão | Data |
| 1 | 2024/01/06 |
|  |  |
|  |  |
|  |  |
|  |  |

**Escola Superior de Tecnologia**

**Licenciatura em Engenharia de Sistemas Informáticos - PL**

Barcelos, 06 de janeiro de 2024

**Índice**

[1. Contextualização 5](#_Toc155554245)

[1.1. Diagrama de Casos de Uso 5](#_Toc155554246)

[1.2. Digrama Entidade – Relação 6](#_Toc155554247)

[2. Introdução do problema 6](#_Toc155554248)

[3. Tecnologias utilizadas 7](#_Toc155554249)

[3.1. PostgreSQL 7](#_Toc155554250)

[3.2. C#, .net e *Entity Framework* 7](#_Toc155554251)

[3.2.1. C# 7](#_Toc155554252)

[3.2.2. .NET 7](#_Toc155554253)

[*3.2.3.* *Entity Framework* 8](#_Toc155554254)

[3.3. Amazon Web Services 8](#_Toc155554255)

[3.4. Node-Red e MQTT 8](#_Toc155554256)

[4. Objetivos 9](#_Toc155554257)

[5. Arquitetura DDD (Domain Driven Design) 9](#_Toc155554258)

[5.1. API 10](#_Toc155554259)

[5.2. Aplicação 11](#_Toc155554260)

[5.3. Infraestrutura 12](#_Toc155554261)

[5.4. Domínio 13](#_Toc155554262)

[6. Regras de Negócio 16](#_Toc155554263)

[6.1. Autenticação de Hóspedes 16](#_Toc155554264)

[6.2. Registo de Acessos 16](#_Toc155554265)

[6.3. Controlo de Acesso Físico e Climatização 16](#_Toc155554266)

[6.4. Interação entre Funcionalidades e Utilizadores 16](#_Toc155554267)

[6.5. Regras para o Check-in e Check-out 16](#_Toc155554268)

[7. JSON Web Token 17](#_Toc155554269)

[8. Documentação 18](#_Toc155554270)

[9. Conclusão 19](#_Toc155554271)

**Índice de Imagens**

[Figura 1 - Diagrama de Casos de Uso 5](#_Toc155554231)

[Figura 2 - Diagrama Entidade – Relação 6](#_Toc155554232)

[Figura 3 - Demonstração da arquitetura 10](#_Toc155554233)

[Figura 4 - Demonstração da camada da API 10](#_Toc155554234)

[Figura 5 - Demonstração da camada da Aplicação 11](#_Toc155554235)

[Figura 6 - Exemplo de um Controller com um dos Endpoints 11](#_Toc155554236)

[Figura 7 - Exemplo de uma chamada retornando para o utilizador 12](#_Toc155554237)

[Figura 8 - Classes com as exceptions 12](#_Toc155554238)

[Figura 9 - Camada Infraestrutura 13](#_Toc155554239)

[Figura 10 – Constituição da camada de domínio 14](#_Toc155554240)

[Figura 11 - Constituição de uma entidade 14](#_Toc155554241)

[Figura 12 - Constituição de uma das classes 15](#_Toc155554242)

[Figura 13 - Classe do TokenGenerator 17](#_Toc155554243)

[Figura 14 - Documentação Swagger 18](#_Toc155554244)

# Contextualização

O projeto integra a matéria lecionada nas Unidades Curriculares (UC) de Projeto Aplicado (PA), Integração de Sistemas de Informação (ISI) e Programação de Dispositivos Móveis (PDM).

Posto isto, o projeto apresentado consiste na criação de um sistema que permita controlar os acessos a quartos de hotéis e controlo da temperatura do respetivo quarto. Para além disso, o sistema também irá permitir consultar o histórico de temperaturas e de acessos aos quartos. O controlo da temperatura e acesso ao quarto será feito através de uma aplicação móvel para Android e o meio físico será controlado através de um Arduino.

## Diagrama de Casos de Uso

O Hóspede pode abrir ou trancar o quarto, autenticar-se na aplicação e alterar a temperatura.

O Serviço de Quarto pode apenas abrir o quarto através do cartão físico.

A receção fornece o cartão físico e insere os dados de acesso à aplicação na base de dados.

Uma imagem com texto, diagrama, captura de ecrã, file

Descrição gerada automaticamente

Figura - Diagrama de Casos de Uso

## Digrama Entidade – Relação

Para o funcionamento do sistema, de forma a guardar o IP de cada controlador de quarto (Arduino) para depois fornecer à aplicação são necessárias as seguintes tabelas.

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura - Diagrama Entidade – Relação

# Introdução do problema

O projeto da UC de ISI contempla a criação de uma API que irá fazer a gestão de acessos a um controlador de quartos de hotel cujos endpoints serão consumidos por uma aplicação a ser desenvolvida na UC de PDM. Esse controlador por sua vez irá controlar o acesso aos quartos e a climatização do quarto.

# Tecnologias utilizadas

## PostgreSQL

O PostgreSQL atua como o sistema central de armazenamento de dados, ou seja, uma base de dados. Este sistema permite guardar e gerir informações essenciais, como informações sobre o quarto, dados das reservas e históricos de acessos.

A sua escolha deve-se à fiabilidade e robustez, assegurando a integridade dos dados e a eficácia na gestão das relações entre diferentes componentes do sistema, como hóspedes, quartos e serviços.

## C#, .net e *Entity Framework*

### C#

C# é uma linguagem de programação orientada a objetos desenvolvida pela Microsoft. É uma linguagem robusta e versátil que se integra bem com o ecossistema .NET. No contexto deste projeto, C# foi a linguagem escolhida para desenvolver a lógica de negócios, a API e outras componentes essenciais do sistema.

### .NET

O .NET, desenvolvido pela Microsoft, é uma framework abrangente para o desenvolvimento, proporcionando uma variedade de bibliotecas e ferramentas. Ela é fundamental para criar aplicações bem estruturadas e de alto desempenho e seguras. A escolha do .NET para este projeto visou garantir uma integração entre os diferentes componentes, resultando numa solução compacta e eficiente. Além disso, a framework oferece recursos avançados, incluindo segurança aprimorada, conectividade eficiente com bancos de dados e muito mais, todos essenciais para elevar a qualidade e solidez da aplicação em desenvolvimento.

### *Entity Framework*

O *Entity Framework* (EF) é um ORM (*Object-Relational Mapping*) desenvolvido pela Microsoft que permite aos desenvolvedores trabalhar com bancos de dados relacionais usando objetos e propriedades em C#. O EF simplifica a interação com o banco de dados, permitindo a criação, leitura, atualização e exclusão de registos de forma intuitiva e eficiente, sem a necessidade de escrever consultas SQL complexas.

## Amazon Web Services

Para atender aos requisitos do projeto, a API e base de dados inerente a ela foram colocadas num servidor Ubuntu Server na Amazon Web Services (AWS).

O Ubuntu Server alojado na AWS permite ajustar recursos conforme necessário. Isso significa que podemos adaptar capacidades como armazenamento e memória da máquina virtual conforme o projeto evolui.

A escalabilidade na AWS garante que o sistema pode lidar com picos de tráfego sem problemas. Num caso de mundo real se a aplicação fosse colocada em distribuição se houvesse um aumento na demanda, a infraestrutura pode ser facilmente ajustada para acomodar mais utilizadores.

Quanto à segurança ela também oferece ferramentas robustas como a utilização de firewalls e encriptação. Podemos proteger a API e a base de dados de ameaças externas, garantindo que os dados permaneçam seguros.

## Node-Red e MQTT

Tendo a autenticação sido feita na aplicação através da API, resolvemos aplicar outras tecnologias que foram lecionadas nas aulas que eram novidade para nós, nomeadamente Node-Red e MQTT.

Essas tecnologias foram utilizadas primariamente nas UC de PDM e de SETR, mas tendo em conta que foram lecionadas nesta UC, consideramos pertinente a sua inclusão neste relatório.

O Node-Red neste caso está instalado localmente nos computadores onde estão a ser desenvolvidos os projetos de PDM e SETR e nele está a ser utilizado uma extensão que permite criar um servidor ou *broker* MQTT dentro do próprio Node-Red.

Posto isto, todas as ações relativas à comunicação entre a App e o Arduino funcionam sobre mensagens MQTT.

# Objetivos

Pretende-se que a API torne possível os seguintes cenários:

Na criação de uma reserva a API após a inserção dos dados da reserva, irá devolver o número da reserva e a password para acesso à aplicação.

Ao aceder na aplicação com os dados previamente fornecidos, a API irá devolver um token e os detalhes da reserva por exemplo o número do quarto.

Também é possível fazer inserções na base de dados através da API que neste caso será utilizada para adicionar registos na tabela de histórico de acessos aos quartos.

# Arquitetura DDD (Domain Driven Design)

Escolhemos a arquitetura DDD pois ela consiste numa abordagem de design de software concentrada na compreensão do domínio da aplicação e na modelação deste domínio no código.

Alguns conceitos que envolvem essa arquitetura são as entidades, *value objects*, agregados, repositórios, serviços de domínio e eventos de domínio.

O DDD visa criar um modelo rico e compartilhado entre equipas de desenvolvimento. A abordagem também incorpora conceitos como *Ubiquitous Language* (Linguagem Ambígua) para garantir que todos os envolvidos partilhem a mesma compreensão do domínio.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Demonstração da arquitetura

## API

Dentro desta arquitetura a API é uma parte da camada de aplicação, que é responsável por expor as funcionalidades do domínio para o mundo externo.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Demonstração da camada da API

## Aplicação

A camada de aplicação atua como uma interface entre o domínio e os utilizadores.

Possui serviços de aplicação responsáveis por organizar as chamadas realizadas ao domínio.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura - Demonstração da camada da Aplicação

Também utiliza de interfaces onde estão alocados os *endpoints* onde ocorrem as chamadas HTTP, recebendo dados das solicitações, validando e convertendo em funções ou consultas para o domínio.

Uma imagem com texto, software, Software de multimédia, Software gráfico

Descrição gerada automaticamente

Figura - Exemplo de um Controller com um dos Endpoints

Quando o domínio é chamado, a aplicação realiza as operações onde a lógica de negócios é encapsulada em entidades, agregados e serviços.

Após isto, ela retorna as operações convertendo os resultados em formatos adequados a resposta esperada de forma que o usuário final tenha o que foi solicitado.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Exemplo de uma chamada retornando para o utilizador

Também na camada de aplicação ocorrem os tratamentos de erros e exceções, para que caso ocorra alguma falha, ela seja relatada de forma amigável para o utilizador final e que ele consiga posteriormente reportar o erro de forma a ser facilmente compreensível pelo suporte.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Classes com as exceptions

## Infraestrutura

Esta camada abrange componentes que lidam com detalhes técnicos como persistência de dados, comunicação com serviços externos e gestão de recursos.

Possui também repositórios e provedores de dados responsáveis por interagir com a base de dados, encapsulando operações como leitura e escrita.

A comunicação externa é feita através de módulos para interação com serviços externos como por exemplo o consumo de APIs de terceiros ou sistemas legados.

A gestão de recursos lida com aspetos relacionados a conexões de rede, e recursos do sistema e elementos técnicos necessários para execução da aplicação.

Também trata da configuração da aplicação, gestão de informações sensíveis e garantia de práticas seguras.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Camada Infraestrutura

## Domínio

A essência da aplicação encontra-se nesta camada. Nela ficam as regras de negócio e as representações do mundo real, como as entidades, agregados e serviços.

As entidades consistem em objetos com identidade própria que refletem conceitos significativos no domínio da aplicação.

Os agregados reúnem agrupamentos de entidades e objetos de valor de forma a garantir consistência e integridade.

Serviços de domínio são componentes que contém regras de negócio, mas que, entretanto, não se encaixam diretamente as entidades ou aos agregados.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura – Constituição da camada de domínio

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Constituição de uma entidade

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Constituição de uma das classes

# Regras de Negócio

## Autenticação de Hóspedes

A regra de negócio estabelece que o hóspede deve autenticar-se na aplicação através do número do quarto e password, sem necessidade de registo na aplicação. Estes dados serão armazenados numa base de dados e serão alterados ou apagados após a estadia do hóspede.

## Registo de Acessos

Estabelece-se que será mantido um registo detalhado de todos os acessos aos quartos incluindo data e hora.

## Controlo de Acesso Físico e Climatização

Estabelece-se a regra de que os hóspedes podem controlar o acesso físico aos quartos e a climatização do mesmo através da aplicação móvel.

## Interação entre Funcionalidades e Utilizadores

Define como as diferentes funcionalidades são acionadas pelos diferentes utilizadores (Hóspede, Serviço de Quarto e Rececionista). Por exemplo, o Serviço de Quarto pode abrir o quarto apenas com um cartão físico, enquanto o hóspede tem a opção de utilizar a aplicação móvel.

## Regras para o Check-in e Check-out

O / A rececionista é responsável por fornecer um cartão ao hóspede para aceder ao quarto e as credenciais de acesso à app e inserir todos os dados necessários no sistema para registar o hóspede. Este cartão e as credenciais de acesso à aplicação são associados ao quarto atribuído ao hóspede.

# JSON Web Token

O JSON Web Token (JWT) é uma forma segura e compacta de enviar informações entre diferentes partes. Funciona como um tipo de "credencial" que contém informações e é usado para verificar se alguém é quem diz ser.

No projeto o JWT garante a segurança dos pedidos feitos à API por exemplo quando os hóspedes querem entrar na app, usam um número de quarto e uma senha. Depois, recebem um token. Com base nisso, todos os pedidos http subsequentes incluem o token que foi fornecido e caso esse token não seja válido, o pedido é rejeitado.

Uma imagem com texto, software, Software de multimédia, Software gráfico

Descrição gerada automaticamente

Figura - Classe do TokenGenerator

# Documentação

Para documentar a API, mais uma vez de forma a corresponder ao enunciado do trabalho, recorremos ao Swagger. Com o Swagger temos de certa forma um guia prático que mostra de forma interativa como é que a API funciona e quais são os seus endpoints.

O Swagger é principalmente útil pois permite fazer testes diretamente nela como se ele se tratasse do Postman, mas de forma mais intuitiva.

A documentação pode ser consultada no seguinte link: <http://13.53.36.123/swagger/index.html>

Uma imagem com texto, captura de ecrã, software, Ícone de computador

Descrição gerada automaticamente

Figura - Documentação Swagger

# Conclusão

Consideramos que o nosso grupo ao realizar o projeto as ferramentas necessárias para tornar realidade o projeto proposto em PA, demonstrou proficiência ao aplicar um conjunto diverso de tecnologias.

A escolha do PostgreSQL assegurou a fiabilidade na gestão de detalhes essenciais como quartos e históricos de acessos. Com C# e .NET, desenvolvemos a lógica de negócios, enquanto o Entity Framework facilitou as interações com a base de dados. Optamos pela AWS, especificamente um servidor Ubuntu, destacando nosso foco em escalabilidade e segurança, reforçado por ferramentas como firewalls. Utilizamos Node-Red e MQTT para eficaz comunicação com o Arduino, demonstrando nossa adaptabilidade. A segurança foi reforçada pelo uso de JSON Web Token, garantindo acessos autorizados, e a documentação foi otimizada através do Swagger, simplificando testes e validações.

Em resumo, através da combinação estratégica e eficaz destas tecnologias, podemos afirmar que o nosso grupo entregou uma solução integrada, robusta e segura, evidenciando competência e aplicação prática dos conhecimentos adquiridos ao longo do semestre na unidade curricular de Integração de Sistemas de Informação.