23 de dezembro de 2022

Renato Silva (23511)

Mário PINTO (23506)

Rúben COSTA (23508)

Trabalho Prático

Programação orientada a objetos

Conteúdo

[**Visão do Projeto** 4](#_Toc122711744)

[**Descrição do Problema** 4](#_Toc122711745)

[**Objetivos e os benefícios deste projeto** 4](#_Toc122711746)

[**Diagrama de Use Cases** 5](#_Toc122711747)

[**Project SmarPark** 6](#_Toc122711748)

[**Project SmarPark.Models** 6](#_Toc122711749)

[Class Movement.cs 7](#_Toc122711750)

[Class Parque.cs 8](#_Toc122711751)

[Class Reserva.cs 9](#_Toc122711752)

[Class Siga.cs 10](#_Toc122711753)

[Class User.cs 11](#_Toc122711754)

[Class Veiculo.cs 12](#_Toc122711755)

[**SmartPark.Enums** 13](#_Toc122711756)

[Enum TipoEstatuto 13](#_Toc122711757)

[**SmartPark.Exceptions** 14](#_Toc122711758)

[Exception EmailDoesntExistException 14](#_Toc122711759)

[**SmartPark.Data** 15](#_Toc122711760)

[Movements.cs 15](#_Toc122711761)

[Criação do Public void Add(Movement newMovement) 16](#_Toc122711762)

[Criamos este método GetMovements() 16](#_Toc122711763)

[Criamos este método GetMovementsbyPlate() 17](#_Toc122711764)

[Criamos este método GetMovementsbyDay() 17](#_Toc122711765)

[Parques.cs 18](#_Toc122711766)

[Criação do Public void Add(Parque newParque) 19](#_Toc122711767)

[Criamos este método GetParques() 19](#_Toc122711768)

[public void Remove(int parqueID) 20](#_Toc122711769)

[public void UpdateSpaces(int parqueId, int spaceValue) 20](#_Toc122711770)

[public bool ExistsAvailableSpaces(int parqueId) 21](#_Toc122711771)

[Reservas.cs 22](#_Toc122711772)

[public void Add(Reserva newReserva, Users users, Parques parque) 22](#_Toc122711773)

[Criamos este método GetReservas() 23](#_Toc122711774)

[Users.cs 23](#_Toc122711775)

[Criação do Public void Add(User newUser) 24](#_Toc122711776)

[Public bool ValidatePasswor() 24](#_Toc122711777)

[Public int GetGrauUser() 25](#_Toc122711778)

[public void Delete(int userID) 25](#_Toc122711779)

[public bool ExistsEmail(string email) => users.Exists(u => u.Email == email); 26](#_Toc122711780)

[public List<User> GetUserByState(TipoEstatuto tipoEstatuto) 26](#_Toc122711781)

[public int GetIdUserByEmail(string email) 26](#_Toc122711782)

[Veiculos.cs 27](#_Toc122711783)

[public void Add(Veiculo newVeiculo) 27](#_Toc122711784)

[public void Update(Veiculo veiculo) 28](#_Toc122711785)

[public void Delete(string matricula, int IdUser) 28](#_Toc122711786)

[public List<Veiculo> GetUserVeiculos(int idUser) 29](#_Toc122711787)

[public int TotalVeiculos() 29](#_Toc122711788)

[**Projeto.Test** 30](#_Toc122711789)

[**Form1** 30](#_Toc122711790)

[Criação de variáveis publicas e criação de Listas 30](#_Toc122711791)

[Criação do button1 31](#_Toc122711792)

[Criação do button4 31](#_Toc122711793)

[Criação do button5 32](#_Toc122711794)

[Criação do button6 32](#_Toc122711795)

[Criação do bt\_ListarUsers\_Click 33](#_Toc122711796)

[Criação do bt\_DeleteUsers\_Click 34](#_Toc122711797)

[Criação do button9 34](#_Toc122711798)

[Criação do button10 35](#_Toc122711799)

[Criação do btGetMovbYDay 35](#_Toc122711800)

[Criação do button9 36](#_Toc122711801)

[Criação do btGetUsersByState 37](#_Toc122711802)

[Criação do btTotalVeiculos 37](#_Toc122711803)

[**Form2** 38](#_Toc122711804)

[Criação do btLogin 38](#_Toc122711805)

[**InsertReserva.cs** 39](#_Toc122711806)

[Criação do button1 39](#_Toc122711807)

[**InsertMovement.cs** 39](#_Toc122711808)

[Criação do button1 39](#_Toc122711809)

[InsertParque 40](#_Toc122711810)

[Criação do button 1 40](#_Toc122711811)

[RegisterVeiculo 41](#_Toc122711812)

[Criação do button 1 41](#_Toc122711813)

[RegisterUser 42](#_Toc122711814)

[Criação do btRegisted 42](#_Toc122711815)

[**Conclusão** 43](#_Toc122711816)

**Conclusão…………………………………………………………………………………………………………………………….9**

# **Visão do Projeto**

O projeto tem como objetivo melhorar e inovar os parques de estacionamento do IPCA. Passará por arranjar uma solução para os problemas em estacionar dos alunos, professores e todas as pessoas pertencentes ao IPCA. Decidimos desenvolver todo um projeto em que fosse possível existir uma visualização das vagas nos parques, tal como acontece nos parques de estacionamento dos centros comerciais e supermercados.

# **Descrição do Problema**

Devido ao resultado de um questionário feito no IPCA juntamente com a nossa experiência em estacionar no IPCA, denotamos que os parques estavam a ser mal geridos. A administração pretende criar um sistema que fosse possível existir uma visualização das vagas nos parques, de forma a reduzir o tempo de procura de estacionamento e evitar o atraso às aulas.

Isto gera diversos problemas, para isso tem de ser criadas as condições necessárias, tais como:

* Sensores para entradas e saídas dos parques
* Câmara para identificar a matrícula
* barreiras para um melhor controlo dos carros que entram no parque
* Aprovação e colaboração do IPCA.

Dito isto, iriamos oferecer aos alunos uma aplicação móvel ou via website que daria acesso as vagas existentes dentro de cada parque assim, no caso de não haver nenhum lugar disponível o aluno não gastaria combustível a procurar. Para registarmos as entradas e saídas dos carros utilizaríamos um sensor em cada zona dos parques em todo o campus IPCA. Na entrada de cada parque será lida as matrículas dos carros. o carro será introduzido na base de dados para saber em que zona do parque o carro entrou, juntamente com as horas de entrada. Na saída de cada parque teremos outro sensor onde registaria na base de dados outro registo de saída do parque com a matrícula e a respetiva hora do acontecimento.

# **Objetivos e os benefícios deste projeto**

O nosso objetivo neste projeto consistirá em criar uma divisão de todo o parque do IPCA em 3, onde cada parque, preferencialmente o mais perto de cada polo, terá o nome de EST, ESG, ESD. Em cada parque indicará o número de vagas disponíveis do mesmo e vai sempre sugerir ao utilizador o parque mais perto do polo que ele frequenta, mas não o irá proibir de estacionar em qualquer um dos outros parques.

* Evitar atrasos às aulas
* reduzir o transito no Campus
* Obter uma melhor gerência dos parques do Campus

# **Diagrama de Use Cases**

Diagram

Description automatically generated

# **Project SmarPark**

Neste projeto criamos 6 projetos em que cada tem os seus objetivos e que são ligados atraves de dependencias, neste O models depende dos enums, o data depende dos models dos enums e das exceptions.

# **Project SmarPark.Models**

Dentro do projeto SmartPark.Models estão inseridas todas as classes com as respetivas propriedades utilizadas ao longo de todo projeto, e em conjunto com os seus contrutores para organização de dados durante a criação de novos objetos.

A screenshot of a computer

Description automatically generated with medium confidence

## Class Movement.cs

Class Movement, esta classe contem 5 propriedades publicas, em que o idMovimento é auto incrementado, para ser auto incrementado foi criada uma variavel estatica com o valor de inteiro (“idIncrementer”), em que no construtor sempre que for criado um novo objeto adiciona sempre +1. Criamos um construtor para durante a criação de objetos, eles serem criados e organizados por uma ordem certa.

Text

Description automatically generated

Text

Description automatically generated

## Class Parque.cs

Class Parque, esta classe contem 4 propriedades publicas, em que o ParqueID é auto incrementado, para ser auto incrementado foi criada uma variavel estatica com o valor de inteiro (“idIncrementer”), em que no construtor sempre que for criado um novo objeto adiciona sempre +1. Criamos um construtor para durante a criação de objetos, eles serem criados e organizados por uma ordem certa.

Text

Description automatically generated

Text

Description automatically generated

## Class Reserva.cs

Class Reserva, esta classe contem 4 propriedades publicas, em que o ReservaId é auto incrementado, para ser auto incrementado foi criada uma variavel estatica com o valor de inteiro (“idIncrementer”), em que no construtor sempre que for criado um novo objeto adiciona sempre +1. Criamos um construtor para durante a criação de objetos, eles serem criados e organizados por uma ordem certa.

Text

Description automatically generated

Text

Description automatically generated

## Class Siga.cs

Class Siga, esta classe contem 2 propriedades publicas, em que ambas são propriedades de tipo enum. Criamos um construtor para servir de base ao fazer hierarquia com a classe de modelo de user.

Text

Description automatically generated

## Class User.cs

Class User, esta classe contem 3 propriedades publicas e 2 propriedades privadas, criamos uma variavel do tipo estatica com o nome de idIncrementer, esta variavel utilizados para igual ao idUser e sempre que for criado um novo objeto vai sempre adicionar +1 ao idUser. Como a password e o NivelUser são propriedades privadas criamos public string Password para dar return da password, retornada do public string GetPassword e criamos tambem um public int NivelUser que vai dar return do Grau do utilizador ao ser criado um objeto. Criamos um construtor para ao ser instanciado um novo objeto ter uma devida ordem de inserção de dados.

Text

Description automatically generatedText

Description automatically generated

## Class Veiculo.cs

Class Veiculo, esta classe contem 3 propriedades publicas. Criamos um construtor para durante a criação de objetos, eles serem criados e organizados por uma ordem certa.

Text

Description automatically generated

# **SmartPark.Enums**

Criamos o SmartPark Enums é aqui onde são guardados os 4 tipos de enumeradores que utilizamos ao longo do projeto completo (“TipoEstatuto.cs”),(“TipoMovimento.cs”),(“TipoPolo.cs”),(“TipoVeiculo.cs”)

Text

Description automatically generated

## Enum TipoEstatuto

Aqui encontra se a enum de TipoEstatuto, onde criamos três opções de recolha de dados em que o utilizador so existe estes três tipos de estatuto, não permitindo ao utilizador escolher um valor diferente destes três e o mesmo se aplica ao restantes enums.

Text

Description automatically generated

# **SmartPark.Exceptions**

Criamos o SmartPark.Exceptions é aqui onde são guardados as varias exceções que utilizamos ao longo de todo o projeto, para serem aplicados nas verificações dos métodos associados as respetivas exceções.

Text

Description automatically generated

## Exception EmailDoesntExistException

Criamos esta exception para enviar esta mensagem caso o email de utilizador não exista e as outras exceptions tem a mesma base de desenvolvimento com objetivos diferentes.

Text

Description automatically generated

# **SmartPark.Data**

Criamos o SmartPark.Data neste projeto é onde tão guardados todos os métodos utilizados ao longo do projeto completo, neste projeto utiliza os dados vindos de outros projetos através de dependências.

Text

Description automatically generated

## Movements.cs

Criação da List<Movement> movements = new List<Movement>()

Através da classe de modelo Movement que nos permite assim ter uma lista para inserir os movimentos.

Criação do Public Movements ()

Construtor que contem um conjunto de objetos instanciados, para ao ser iniciado o projeto já termos dados inseridos dentro da lista.

Graphical user interface, text

Description automatically generated

### Criação do Public void Add(Movement newMovement)

Criamos este metodo para inserção de novos movimentos, com um parametro newMoviment que provem da classe Movement, neste metodo caso o newMoviment seja diferente de nulo então atraves de um parametro é criado um novo movimento e isto dentro de um try and catch, em que ele tenta criar se conseguir adiciona o novo movimento se não da catch da Exceção MovementNullException.

Text

Description automatically generated

### Criamos este método GetMovements()

Neste método são retornados todos os movements para um array Movement que depois utilizamos para os listarmos.

Text

Description automatically generated

### Criamos este método GetMovementsbyPlate()

Neste método são retornados todos os movements associados a uma matricula e retorna esses movements para o array Movement.

Text

Description automatically generated

### Criamos este método GetMovementsbyDay()

Neste método são retornados todos os movements associados a um dia escolhido pelo utilizador e retorna esses movements para o array Movement.

Text

Description automatically generated

## Parques.cs

Criação da List<Parque> parques = new List<Parque>();

Através da classe de modelo Parque que nos permite assim ter uma lista para inserir os parques.

Criação do Public Parques ()

Construtor que contem um conjunto de objetos instanciados, para ao ser iniciado o projeto já termos dados inseridos dentro da lista.

Graphical user interface, text

Description automatically generated

### Criação do Public void Add(Parque newParque)

Criamos este metodo para inserção de novos parques, com um parametro newParque que provem da classe Parque, neste metodo caso o newParque onde tem varias exceptions, caso sejam todas as exceptions verificadas como corretas, então atraves de um parametro é criado um novo parque e isto dentro de um try and catch, em que ele tenta criar se conseguir adiciona o novo parque se não da catch de Exceção que não foi verificada como correta.

Text

Description automatically generated

### Criamos este método GetParques()

Neste método são retornados todos os parques para um array Parque que depois utilizamos para os listarmos.

Text

Description automatically generated

### public void Remove(int parqueID)

Utilizamos este metodo para eliminar parques, atraves do parametro parqueID.

Text

Description automatically generated

### public void UpdateSpaces(int parqueId, int spaceValue)

Metodo para retornar o update do total de lugares disponiveis por parque.

Text

Description automatically generated

### public bool ExistsAvailableSpaces(int parqueId)

Metodo bool onde retorna um resultado de true or false para verificar se existem lugares disponiveis de um parque especifico caso o numero de lugares disponiveis for diferente de 0 da return true, caso seja igual a zero da um retorno de false.

A screenshot of a computer

Description automatically generated with medium confidence

## Reservas.cs

Criação da List<Reserva> reservas = new List<Reserva>();

Através da classe de modelo Reserva que nos permite assim ter uma lista para inserir os reservas.

Criação do Public Reservas ()

Construtor que contem um conjunto de objetos instanciados, para ao ser iniciado o projeto já termos dados inseridos dentro da lista.

Graphical user interface, text

Description automatically generated

### public void Add(Reserva newReserva, Users users, Parques parque)

Criamos este metodo para inserção de novas reservas, com três parametros newRerserva que provem da classe Reserva, users que provem da classe Users e Parques que provem da classe Parques, neste metodo tem varias exceptions, caso sejam todas as exceptions verificadas como corretas, então atraves de um parametro é criado uma novo reserva e isto dentro de um try and catch, em que ele tenta criar se conseguir adiciona uma nova reserva se não da catch de Exceção que não foi verificada como correta.

Text

Description automatically generated

### Criamos este método GetReservas()

Neste método são retornados todas as reservas para um array Reserva que depois utilizamos para as listarmos.

Graphical user interface, text, website

Description automatically generated

## Users.cs

Criação da List<User> users = new List<User>();

Através da classe de modelo User que nos permite assim ter uma lista para inserir os utilizadores.

Criação do Public Users ()

Construtor que contem um conjunto de objetos instanciados, para ao ser iniciado o projeto já termos dados inseridos dentro da lista.

Text

Description automatically generated

### Criação do Public void Add(User newUser)

Criamos este metodo para inserção de novos utilizadores, com um parametro newUser que provem da classe User, neste metodo caso o newUser seja diferente de nulo então atraves de um parametro é criado um novo utilizador e isto dentro de um try and catch, em que ele tenta criar se conseguir adiciona o novo movimento se não da catch da Exceção UserNullException.

Text

Description automatically generated

### Public bool ValidatePasswor()

Metodo bool que verifica o email e a password de utilizador inseridos e verifica se existem na lita de users, caso existem da return true, caso um dos paramatros não exista ou esteja incorreto da return false.

Graphical user interface, text

Description automatically generated

### Public int GetGrauUser()

Metodo que através do parâmetro email tenta encontrar o id do utilizador associado ao email selecionado.

A screenshot of a computer

Description automatically generated with medium confidence

### public void Delete(int userID)

Utilizamos este metodo para eliminar utilizadores, atraves do parametro userID.

Text

Description automatically generated

### public bool ExistsEmail(string email) => users.Exists(u => u.Email == email);

Metodo do tipo bool para verificação do Email, atraves de um parametro email é verificado todos os emails na lista de users caso o email inserido exista na lista e retornado um resultado true, caso contrario e retornado resultado false.

Text

Description automatically generated

### public List<User> GetUserByState(TipoEstatuto tipoEstatuto)

Metodo que retorna a lista dos utilizadores associados a certo estatuto.

Text

Description automatically generated

### public int GetIdUserByEmail(string email)

Metodo para mediante o email inserido pelo utilizador e encontrado o id associado a esse email e retorna o valor desse id associado.

Text

Description automatically generated

## Veiculos.cs

Criação da List<Veiculo> veiculos = new List<Veiculo>();

Através da classe de modelo Veiculo que nos permite assim ter uma lista para inserir os veiculos.

Criação do Public Veiculos ()

Construtor que contem um conjunto de objetos instanciados, para ao ser iniciado o projeto já termos dados inseridos dentro da lista.

Text

Description automatically generated

### public void Add(Veiculo newVeiculo)

Criamos este metodo para inserção de novos veiculos, com o parametro newVeiculo que provem de Veiculo, neste metodo tem varias exceptions, caso sejam todas as exceptions verificadas como corretas, então atraves de um parametro é criado um novo veiculo e isto dentro de um try and catch, em que ele tenta criar se conseguir adiciona um novo veiculo se não da catch de Exceção que não foi verificada como correta.

Text

Description automatically generated

### public void Update(Veiculo veiculo)

Metodo que permite dar update a um veiculo, para tal executa três exceções primeira se a lista de veiculos esta nula, a segunda verifica se a matricula já existe, e a terceira verifica se o idUserMatricula é única, caso passe pelas exceções da update ao veiculo.

Text

Description automatically generated

### public void Delete(string matricula, int IdUser)

Utilizamos este metodo para eliminar veiculos, atraves do parametro IdUser e matricula.

Text

Description automatically generated

### public List<Veiculo> GetUserVeiculos(int idUser)

Metodo que retorna uma lista de veiculos de um dado User atraves do idUser associado.

Text

Description automatically generated

### public int TotalVeiculos()

Metodo que retorna um total de veículos.

Text

Description automatically generated

# **Projeto.Test**

Conjunto de forms utilizados para testagem da aplicação.

A screenshot of a computer

Description automatically generated with medium confidence

## **Form1**

Aqui é onde está o forms principal de testagem do nosso projeto

### Criação de variáveis publicas e criação de Listas

Variaveis criadas para poder acessar entre os forms, pois sem elas não teríamos acesso as classes presentes no SmartPark.Data

Listas criadas associando a uma variavel

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button1

Dentro deste button conseguimos listar todos os movimentos de uma matricula associada através do método GetMovementsByPlate().

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button4

Dentro deste button conseguimos listar todos os movimentos através do método GetMovementsByPlate().

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button5

Dentro deste button conseguimos remover o parque mediante o Id Recolhido pela input box. Atraves do método Remove().

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button6

Dentro deste button conseguimos obter uma listagem de todos os parques. Atraves do método GetParque().

Uma imagem com texto

Descrição gerada automaticamente

### Criação do bt\_ListarUsers\_Click

Dentro deste button conseguimos obter uma listagem de todos os users. Atraves do método GetUsers().

Uma imagem com texto

Descrição gerada automaticamente

### Criação do bt\_DeleteUsers\_Click

Dentro deste button conseguimos remover o utilizador mediante o Id Recolhido pela input box. Atraves do método Delete().

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button9

Dentro deste button conseguimos obter uma listagem de todos os veículos. Atraves do método veículos.List()

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button10

Dentro deste button conseguimos remover o veiculo através da matricula e id recebido pelo utlizados. Através do método veículos.Delete()

Uma imagem com texto

Descrição gerada automaticamente

### Criação do btGetMovbYDay

Dentro deste button conseguimos obter uma listagem de todos os movimentos realizados naquele dia. Através do método GetMovementByDay()

Uma imagem com texto

Descrição gerada automaticamente

### Criação do button9

Dentro deste button conseguimos obter a listagem de todas as reservas já efetuadas. Através do método reservas.GetReservas()

Uma imagem com texto

Descrição gerada automaticamente

### Criação do btGetUsersByState

Dentro deste button conseguimos obter uma listagem de todos os Users com um determinado estatuto escolhido na comboBox1 pelo Utilizador. Através do método GetUsersByState()

Uma imagem com texto

Descrição gerada automaticamente

### Criação do btTotalVeiculos

Dentro deste button conseguimos obter uma messageBox com o numero de todos os veículos existentes . Através do método veículos.TotalVeiculos()

Uma imagem com texto

Descrição gerada automaticamente

## **Form2**

Form do Login

### Criação do btLogin

Dentro deste Button conseguimos fazer a comparação mediante o grau do utilizador caso o utilizador seja de grau 1 vai para o form1 que é o do utilizador. Caso contrário irá para o Form3 que é o de visitas.

Uma imagem com texto

Descrição gerada automaticamente

## **InsertReserva.cs**

Forms que nos permite Inserir a reserva

### Criação do button1

Dentro deste button conseguimos inserir uma reserva. Através do método reservas.Add()

Uma imagem com texto

Descrição gerada automaticamente

## **InsertMovement.cs**

### Criação do button1

Dentro deste button conseguimos inserir um movimento através da matrícula, Tipo Movimento e parques disponíveis. Através do método parques.ExistsAvailableSpaces()

Uma imagem com texto

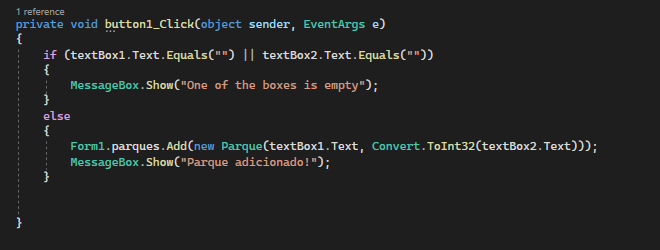
Descrição gerada automaticamente

## InsertParque

Form que nos permite inserir um parque

### Criação do button 1

Dentro deste button conseguimos caso ambas as textboxes não estejam vazias então vão ser inseridos os dados através do método parques.Add().



## RegisterVeiculo

Form para registo de veículos

### Criação do button 1

Dentro deste button conseguimos fazer os registo de veículos associados ao utilizador que se acabou de registar, o utilizador tem a possibilidade de registar mais que um veiculo, para inserir estes veículos a lista utilizamos o método veículos.Add(), e para o associar a um utilizador criamos uma variável estática no form de registar utilizadores que ao se registar ao passar ao form de registo de veículos a aplicação já faz associa já iguala o id do veiculo ao id associado ao utilizador correspondente.

Text

Description automatically generated

## RegisterUser

Form para registo de utilizadores

### Criação do btRegisted

Dentro deste form o utilizador ao clicar no button register, depois de estarem preenchidos todos os campos, o utilizador pode clicar no botão register e cria a sua conta com os dados inseridos esses, são guardados na lista users, através do método users.Add().

Text

Description automatically generated

# **Conclusão**

Em suma, foi um trabalho trabalhoso, mais complicado do que a primeira entrega, algumas coisas poderia ter feito de outra maneira mas em geral penso que apliquei bem os conhecimentos adquiridos ao longo do semestre.