Relatório de Implementação

Sumário: O relatório representa uma visão caracterizada do processo de implementação do sistema desenvolvido. Neste relatório são abordados alguns tópicos como a arquitetura do sistema, principais componentes, fluxo de dados, e o manual de instalação da aplicação. Inclui também a documentação dos testes feitos para validação das funcionalidades, bem como resultados que garantem a fiabilidade do sistema.

Data de preparação: 23/09/2024

Grupo: João Gabriel (Nº 117589)

Luís Assis (Nº 112763)

Luís Nantes (Nº 120401)

Pedro Sampaio (Nº 119213)

Rodrigo Ferreira (Nº 120099)

Versão: V0.1

Circulação: ESTGA-UA

Índice

1 Introdução **5**
2 Arquitetura do sistema **6**
3 Modelo de componentes da solução **7**
3.1 Componentes principais 7
3.1.1 Interface gráfica (GUI) 7
3.1.2 Lógica da aplicação 7
3.1.3 Gestão de dados 8
3.2 Fluxo de dados entre componentes 8
3.2.1 Autenticação e direcionamento 8
3.2.2 Operações específicas 9
3.2.3 Fluxo de resposta 9
4 Instalação **10**
4.1 Manual de instalação 10
5 Testes e validação **16**
5.1 Testes de cifrar 17
5.1.1 Teste de cifrar password válida 17
5.1.2 Teste de cifrar password repetida 17
5.2 Testes em gestão de médicos, consultas, pacientes e horários médicos 17
5.2.1 Teste de adicionar médico 17
5.2.2 Teste de adição e remoção de consulta 17
5.2.3 Teste de adição de paciente 18
5.2.4 Teste de criação e adição de horários médicos 18
5.2.5 Teste de remoção de médico 18
5.2.6 Teste de adição de consultas com pacientes e médicos diferentes 18
5.3 Testes nas consultas 19
5.3.1 Teste de criação de consulta 19
5.3.2 Teste de Getters da consulta 19
5.3.3 Teste de alteração do motivo da consulta 19
5.3.4 Teste de alteração de dados do paciente 19
5.3.5 Teste de alteração do médico da consulta 20
5.4 Testes na Main 20
5.4.1 Teste de inicialização da conexão 20
5.4.2 Teste de visibilidade da janela de login 20
5.4.3 Teste do método Main 21
5.5 Testes no médico 21
5.5.1 Teste de criação de médico 21
5.5.2 Teste de especialidade do médico 21
5.5.3 Teste do número da ordem do médico 21
5.5.4 Teste de criação de horários para o médico 22
5.5.5 Teste de criação de vários horários para o médico 22
5.6 Testes no paciente 22
5.6.1 Teste de criação de paciente 22

5.6.2 Teste dos getters do paciente 22
5.6.3 Teste de igualdade entre pacientes 23
5.6.4 Teste de número SNS padrão 23
5.6.5 Teste de contato padrão 23
5.7 Testes no registo clínico 23
5.7.1 Teste de criação de registo clínico 24
5.7.2 Teste de criação de registo clínico vazio 24
5.8 Testes na interação com a base de dados 24
5.8.1 Teste de criação de paciente 24
5.8.2 Teste de criação de paciente 25
5.8.3 Teste de obter todos os horários dos médicos 25
5.9 Testes em dividir uma string 25
5.9.1 Teste com String válida 25
5.9.2 Teste com String vazia 25
5.9.3 Teste com String nula 26
5.10 Testes em utilizadores 26
5.10.1 Teste de criação de utilizar 26
5.10.2 Teste dos getters do utilizador 26
6 Anexos **28**

1 Introdução

O relatório de implementação descreve os detalhes técnicos e as decisões de implementação tomadas no sistema. Este relatório mostra como foi feita a arquitetura, os componentes principais e os aspetos de implementação utilizados que fazem com que o sistema funcione de forma eficiente.

O sistema foi programado para atender às necessidades de saúde, com funções como autenticação de utilizadores, gestão de consultas, e monitoramento de registos clínicos. Cada módulo foi elaborado de forma modular para certificar a flexibilidade e facilitar futuras melhorias.

O propósito é oferecer um ponto de vista detalhado dos componentes da solução, além de documentar a arquitetura e as fases de desenvolvimento do sistema.

2 Arquitetura do sistema

O sistema foi desenvolvido com uma arquitetura em camadas, separada em:

- **Interface gráfica:** Encarregue de interagir com o utilizador (gestor, funcionário, médico).
- **Lógica da aplicação:** Centra as regras do sistema, como a gestão de consultas e registos clínicos.
- **Gestão de dados:** Gere o acesso e a modificação das informações na base de dados.

3 Modelo de componentes da solução

O modelo de componentes da solução retrata as partes fundamentais que constituem o sistema e como interagem entre si. Essa organização possibilita a divisão de responsabilidades, garante maior modularidade, reutilização e facilidade de manutenção. A estrutura está separada com a interface gráfica, lógica da aplicação e a camada de gestão de dados, cada uma destas divisões representa um papel fundamental no funcionamento do sistema.

3.1 Componentes principais

3.1.1 Interface gráfica (GUI)

- **VistaDeLogin:** Permite o login a utilizadores no sistema e direciona-o para a interface correspondente de cada utilizador (gestor, médico e funcionário).
- **VistaGestor:** Permite ao gestor visualizar, criar e eliminar utilizadores.
- **VistaMedico:** Mostra as consultas associadas ao respetivo médico.
- **VistaSecretaria:** Faz a gestão de consultas, permite marcação, cancelamento e pesquisa de consultas.
- **DisponibilidadeMedicos:** Dentro da vista de secretaria mostra quais os médicos estão disponíveis, e permite fazer uma pesquisa por especialidade.
- **NovaEntradaRC:** Permite fazer um novo registo clínico, ao escrever o assunto e os tratamentos do mesmo.

3.1.2 Lógica da aplicação

- **Utilizador:** Caracteriza os utilizadores do sistema, inclui atributos como o ID, nome, tipo de utilizador (gestor, médico, funcionário).
- **Consulta:** Gestão das informações referentes às consultas, tais como a data, hora, médico e paciente.
- **Paciente:** Retrata os pacientes, armazena dados como o nome, contacto e SNS.
- **Médico:** Tipo de utilizador que adiciona atributos como especialidade e número de ordem.
- **RegistoClinico:** Gestão do histórico médico dos pacientes.

3.1.3 Gestão de dados

- **SqlGeral:** Gestão da conexão com a base de dados e funções gerais, como o login.
- **SqlGestor:** Encarregue da criação e remoção de utilizadores.
- **SqlMedico:** Gere os registos clínicos e as consultas ligadas aos médicos.
- **SqlSecretaria:** Controla as consultas e os pacientes na base de dados.

3.2 Fluxo de dados entre componentes

A comunicação entre os componentes do sistema percorre uma lógica organizada que retrata a arquitetura em camadas. Cada uma destas camadas exerce um papel diferente e interage com as outras de maneira controlada, garante um fluxo eficiente de informações e operações.

A interface gráfica serve como ponto de entrada para os utilizadores, reúne operações como a realização do login, visualização e alteração de dados. Estas operações são realizadas pela lógica da aplicação, esta utiliza as regras definidas e determina as operações que devem ser feitas. Por fim, a gestão de dados é responsável por comunicar com a base de dados para guardar ou recuperar os dados necessários, com um retorno dos resultados para as restantes camadas (interface gráfica, lógica da aplicação).

3.2.1 Autenticação e direcionamento

- O utilizador efetua o seu login através da VistaDeLogin, que chama o método *verificarLogin* situado na SqlGeral.
- Conforme o tipo de utilizador:
- Caso o utilizador seja gestor, acessa a VistaGestor e pode comunicar com o SqlGestor para realizar a gestão dos utilizadores.
- Caso o utilizador seja médico, liga-se à VistaMedico e utiliza o SqlMedico para carregar consultas.
- Caso o utilizador seja um funcionário, este acessa a VistaSecretaria e interage com SqlSecretaria onde efetua a gestão de consultas e pacientes.

3.2.2 Operações específicas

- Cada vista interage com a camada de lógica para serem aplicadas as regras de lógica.
- A camada da lógica chama os componentes de gestão de dados (SqlGeral, SqlGestor, SqlMedico, SqlSecretaria) para que possa acessar ou alterar os dados na base de dados.

3.2.3 Fluxo de resposta

- Os dados serão processados pela camada de gestão de dados e serão retornados posteriormente para a lógica da aplicação.
- A interface gráfica mostra as informações pedidas ao utilizador.

4 Instalação

4.1 Manual de instalação

Ao clicar no link disponibilizado será transferido um arquivo ".rar" diretamente para área download de seus arquivos do PC, após isso descompacte o arquivo, será mostrado um arquivo chamado "MediFlow" como Mostrado na **Figura 1**.

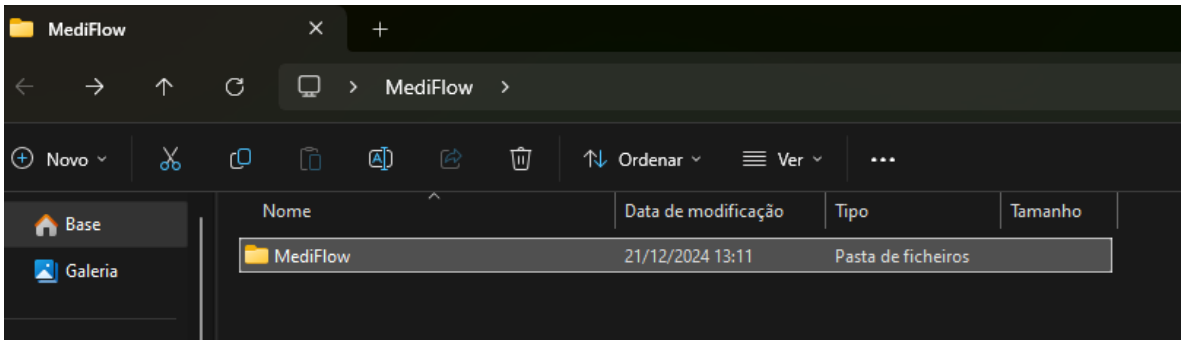

Figura 1

Após isso abra o arquivo e nele estarão mais outros dois arquivos com os nomes de "Installer" e "java" (**Figura 2**), abra o arquivo Installer e nele estará já o executável do installer da aplicação (**Figura 3**).

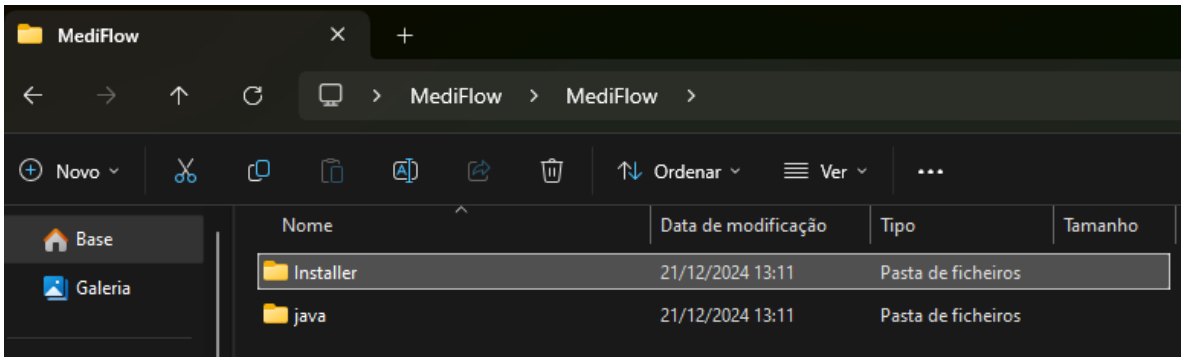

Figura 2

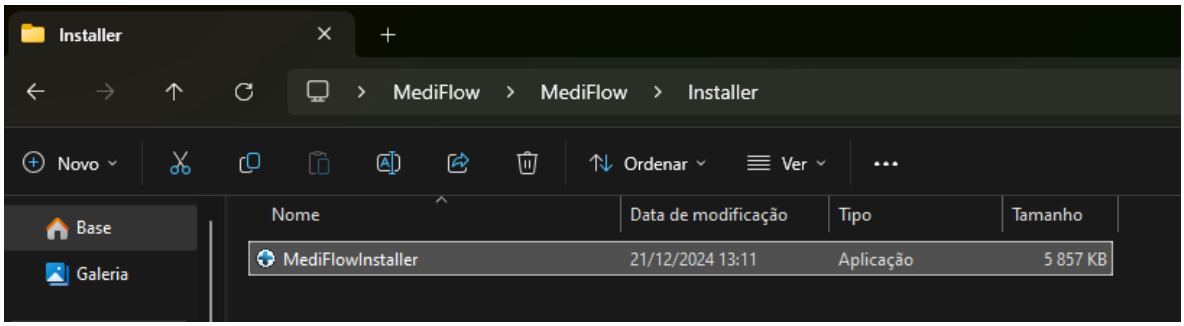

Figura 3

Ao clicar no executável, o PC dará início à instalação da aplicação.

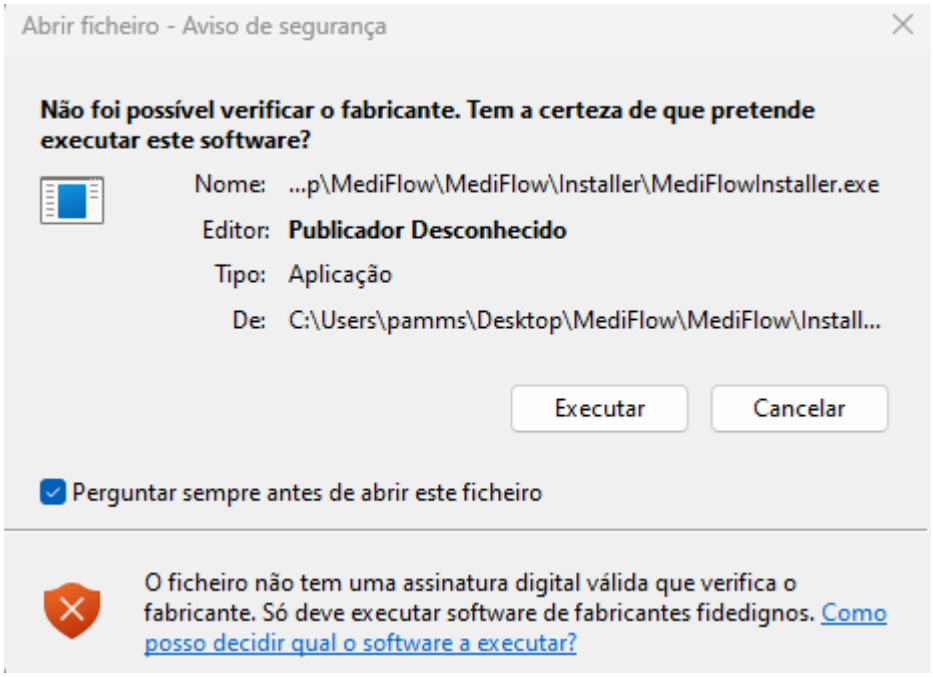

Figura 4

Ao iniciar o installer aparecerá uma tela como está mostrada na **Figura 4**. Clique em executar.

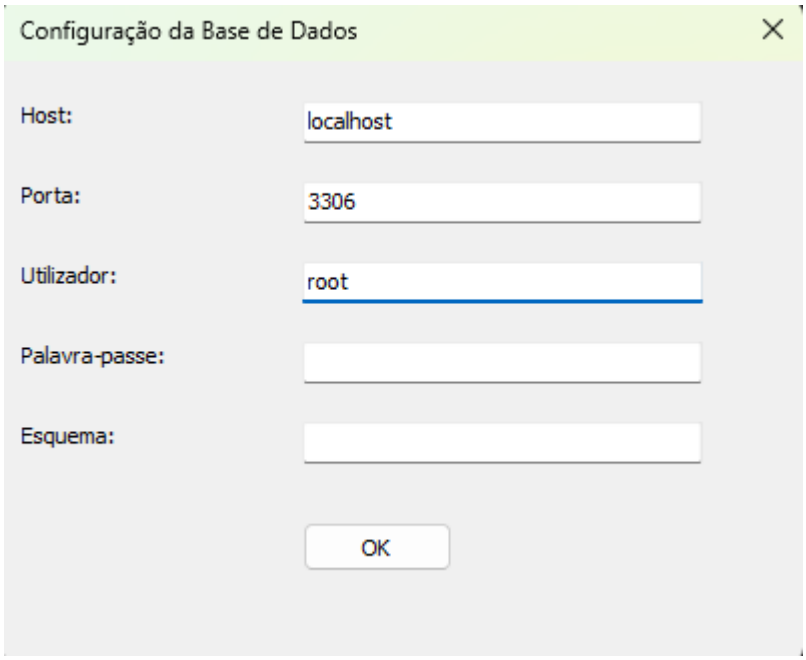

Figura 5

Ao clicar em executar será aberta uma tela como está mostrada na **Figura 5**, insira os dados da Base de Dados em que a aplicação se baseará, após isso clique em "ok", se todos os dados estiverem corretos, você avançará para o próximo passo.

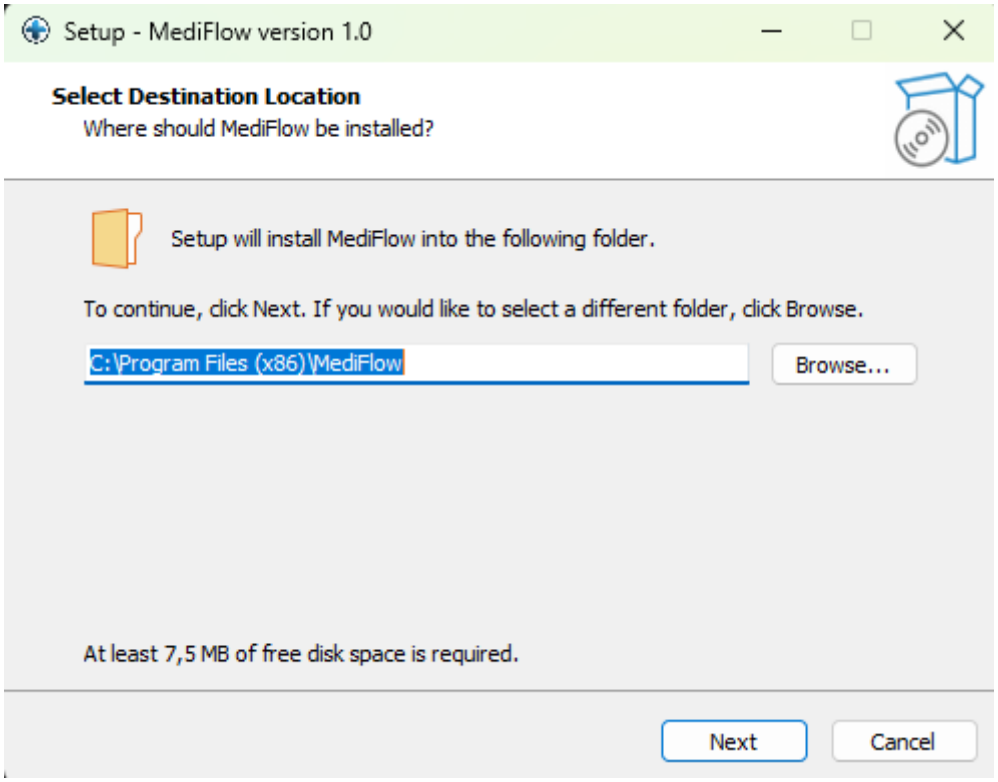

Figura 6

Neste passo, o installer pedirá para que seja criado um arquivo junto aos arquivos de programa como mostrado na **Figura 6**. Caso queira mudar o local de criação de arquivos

clique em “browse...” e escolha outro local. Após a seleção de caminhos para a criação de arquivos, clique em “next”.

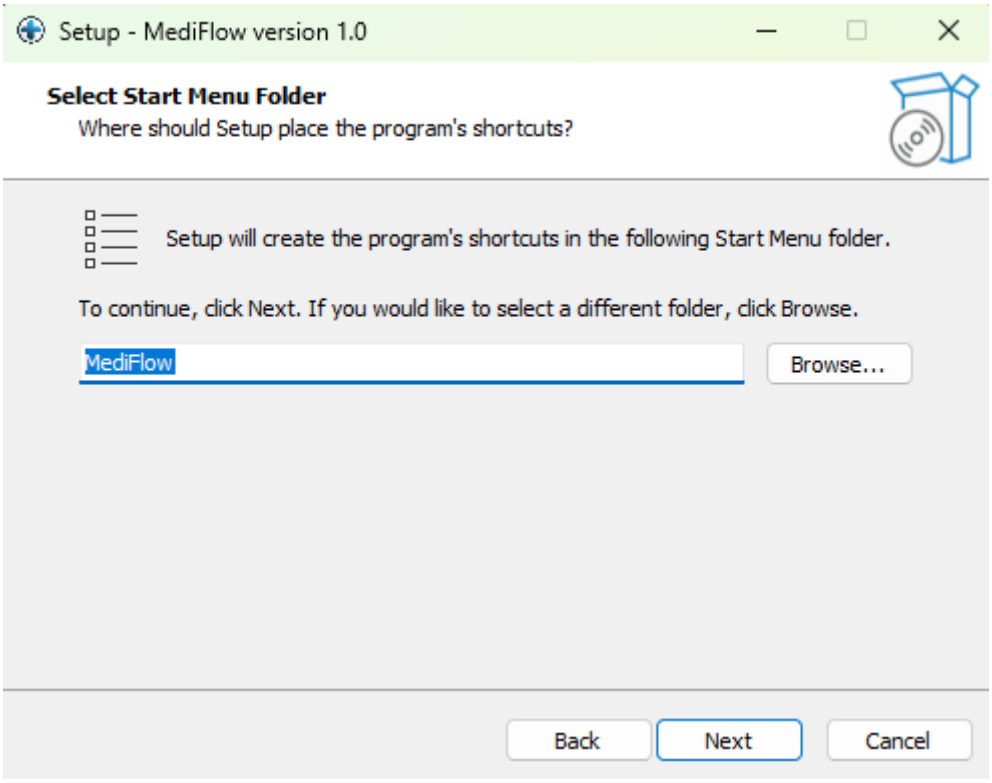

Figura 7

Após clicar em “next”, será aberta esta tela onde o installer irá criar um atalho na Área de Trabalho do seu PC como mostrado na **Figura 7**, caso queira mudar o local de criação de atalho clique em “browse...”, e depois em “next”.

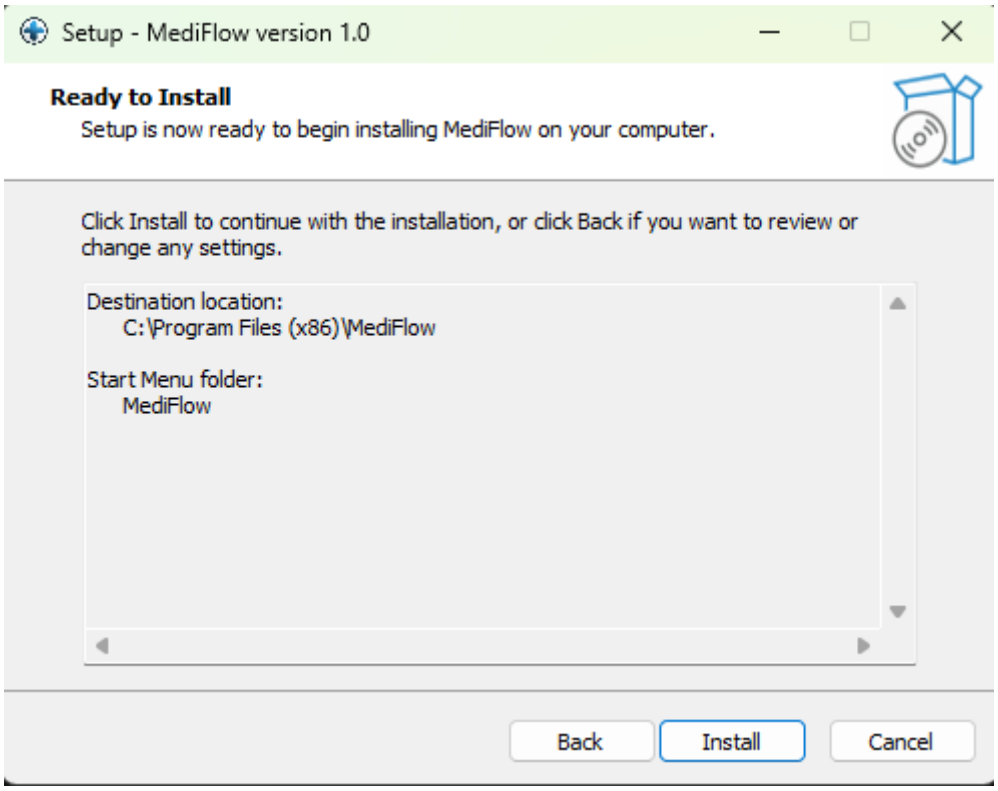

Figura 8

Após todos estes passos o installer já está pronto para fazer a instalação da aplicação, clique em "Install" caso queira já instalar a aplicação.

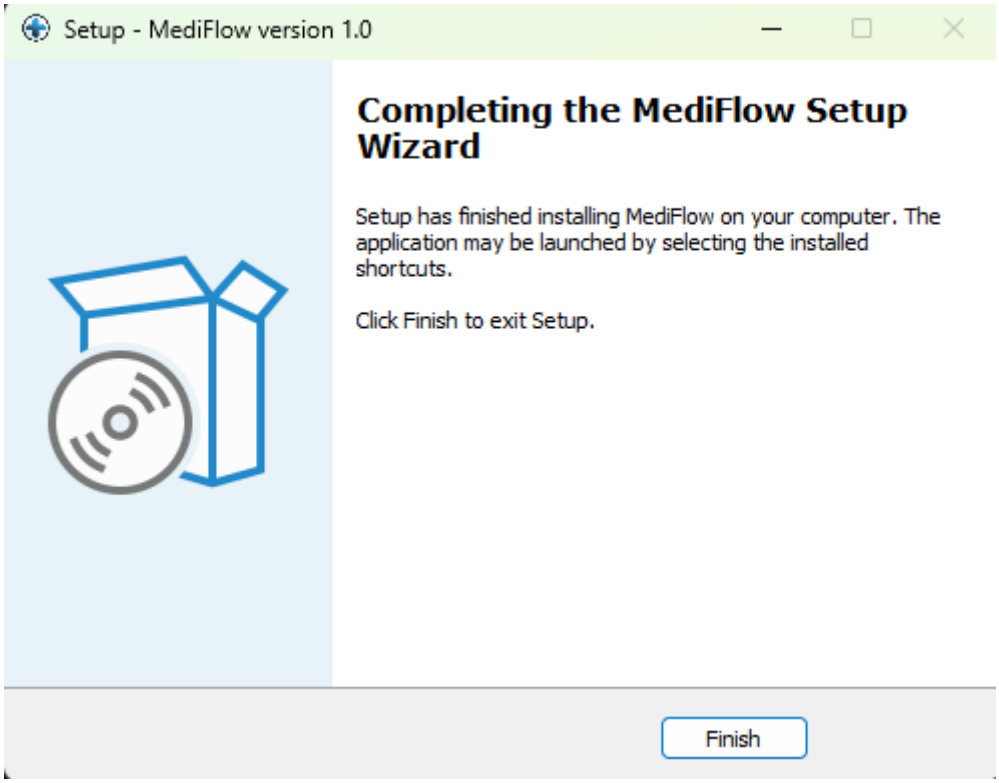

Figura 9

Após todos estes passos a aplicação já foi instalada, clique em "Finish" (mostrada **na Figura 9**) para fechar o installer, caso tenha escolhido que o atalho fosse criado na Área de Trabalho aparecerá um ícone da aplicação como na **Figura 10**.

Figura 10

Observação: Caso a base de dados não esteja configurada corra o script do sql no arquivo sql.

5 Testes e validação

Os testes unitários é uma fase importante no desenvolvimento de software, permitem verificar o desempenho esperado das diferentes funcionalidades implementadas no sistema. Os testes foram realizados para garantir a integridade e fiabilidade das operações principais, identifica também possíveis falhas antes que o sistema seja utilizado em um ambiente real.

Cada funcionalidade foi avaliada em cenários diferentes, estes incluem casos de sucesso e falha, de forma a garantir que o sistema lida corretamente com entradas válidas e inválidas. Posteriormente, serão detalhados os testes unitários desenvolvidos, ao realçar os objetivos, os resultados esperados e os resultados obtidos para cada caso.

5.1 Testes de cifrar

O objetivo destes testes é garantir que as passwords são cifradas de maneira correta.

5.1.1 Teste de cifrar password válida

- **Descrição:** Verifica se é cifrada uma password válida ao receber uma entrada válida (não nula e não vazia).
- **Resultado esperado:** A password cifrada deve ser uma string não nula e não vazia.
- **Resultado obtido:** O teste foi bem sucedido, o método *cifrar* gera resultados válidos para entradas válidas.

5.1.2 Teste de cifrar password repetida

- **Descrição:** Certifica que o método retorna sempre a mesma password cifrada ao receber a mesma entrada.
- **Resultado esperado:** As passwords cifradas devem ser idênticas para a mesma entrada.
- **Resultado obtido:** Teste aprovado, o método *cifrar* funciona corretamente.

5.2 Testes em gestão de médicos, consultas, pacientes e horários médicos

Os testes seguintes têm como objetivo garantir um bom funcionamento das principais funcionalidades da classe Clinica, esta é responsável pela gestão de médicos, consultas, pacientes e horários médicos. Estes testes garantem que as operações de adicionar, remover e manipular dados respeitam os requisitos do sistema e mantém a integridade das informações.

5.2.1 Teste de adicionar médico

- **Descrição:** É adicionado um novo médico à lista de médicos, e verifica se o número de médicos na lista aumenta.
- **Resultado esperado:** A lista de médicos aumenta, ao acrescentar um médico.
- **Resultado obtido:** O teste foi aprovado, confirmou-se que o médico foi criado corretamente e adiciona à lista.

5.2.2 Teste de adição e remoção de consulta

- **Descrição:** É criada uma consulta, é adicionada ao sistema e posteriormente removida. Após esta operação, verifica se o número de consultas na lista mudou.

- **Resultado esperado:** Depois de ser feita a adição da consulta, a lista de consultas deve aumentar um item, e após a remoção, deve diminuir um item.
- **Resultado obtido:** O teste passou com sucesso o teste, confirmou que a adição e remoção de consultas está a funcionar corretamente.

5.2.3 Teste de adição de paciente

- **Descrição:** É criado um paciente, e verifica se o paciente foi registado corretamente.
- **Resultado esperado:** Após esta operação, deve aumentar um item na lista de pacientes.
- **Resultado obtido:** Teste aprovado, o paciente foi adicionado corretamente.

5.2.4 Teste de criação e adição de horários médicos

- **Descrição:** São criados dois horários disponíveis para um médico e verifica se a quantidade de horários criados é consistente.
- **Resultado esperado:** A lista de horários médicos deve conter dois itens.
- **Resultado obtido:** O teste foi aprovado, os horários foram criados corretamente.

5.2.5 Teste de remoção de médico

- **Descrição:** É adicionado um médico e em seguida remove-se esse mesmo médico. Depois desta operação verifica se este médico está na lista de médicos.
- **Resultado esperado:** Após a remoção do médico, a lista não deve conter este médico.
- **Resultado obtido:** Teste bem sucedido, o médico foi removido de maneira correta.

5.2.6 Teste de adição de consultas com pacientes e médicos diferentes

- **Descrição:** São criados dois pacientes, dois médicos e duas consultas. Verifica se as consultas foram registadas corretamente.
- **Resultado esperado:** Após a operação, deve ser acrescentado dois itens à lista de consultas.
- **Resultado obtido:** O teste foi aprovado, as consultas foram adicionadas corretamente.

Os testes implementados mostram que as funcionalidades relacionadas à gestão de médicos, consultas, pacientes e horários médicos estão a ser operadas de forma correta, garantem a integridade e o funcionamento correto do sistema.

5.3 Testes nas consultas

Os testes realizados nas consultas têm como objetivo garantir uma integridade nas operações feitas nas consultas do sistema. Esta classe envolve a criação e validação de dados até as alterações feitas em atributos como médico, paciente e motivo.

5.3.1 Teste de criação de consulta

- **Descrição:** É criada uma consulta e verifica se todos os atributos são inicializados de forma correta.
- **Resultado esperado:** Os atributos da consulta devem coincidir com os valores fornecidos durante a sua criação.
- **Resultado obtido:** O teste foi aprovado, confirma que a consulta é criada corretamente.

5.3.2 Teste de Getters da consulta

- **Descrição:** Certifica que os métodos getters da classe Consulta retornam os valores esperados para os atributos da consulta.
- **Resultado esperado:** Cada método getter deve retornar o valor correspondente ao atributo integrado.
- **Resultado obtido:** O teste teve sucesso, demonstrou que todos os getters funcionam de forma correta.

5.3.3 Teste de alteração do motivo da consulta

- **Descrição:** Altera-se o motivo da consulta e verifica se a alteração é refletida corretamente.
- **Resultado esperado:** O motivo modificado deve ser o novo motivo da consulta atualizado.
- **Resultado obtido:** O teste foi aprovado, confirmou que o motivo foi alterado corretamente.

5.3.4 Teste de alteração de dados do paciente

- **Descrição:** É alterado o paciente associado a uma consulta e verifica se a alteração foi feita corretamente.
- **Resultado esperado:** O nome do paciente associado à consulta deve ser atualizado.
- **Resultado obtido:** O teste foi aprovado, a alteração do paciente foi realizada conforme esperado.

5.3.5 Teste de alteração do médico da consulta

- **Descrição:** É alterado o médico associado a uma consulta e verifica se a mudança foi bem sucedida.
- **Resultado esperado:** O nome do médico associado à consulta deve ser atualizado.
- **Resultado obtido:** O teste foi bem sucedido, a alteração do médico ocorreu corretamente.

Os testes feitos demonstram que as funcionalidades envolvidas com a consulta, tais como a sua criação, alteração de motivo, dados do paciente e médico estão a ser bem sucedidos.

5.4 Testes na Main

Os testes a seguir têm como objetivo validar as operações relacionadas com a inicialização e execução do sistema. Os testes asseguram que o sistema inicializa de forma correta e que os elementos principais, tais como a conexão com a base de dados e a interface gráfica, funcionam corretamente.

5.4.1 Teste de inicialização da conexão

- **Descrição:** Certifica que a conexão com a base de dados é inicializada corretamente.
- **Resultado esperado:** A conexão com a base de dados deve ser estabelecida corretamente e não pode ser nula.
- **Resultado obtido:** O teste foi aprovado, a conexão foi inicializada de forma correta.

5.4.2 Teste de visibilidade da janela de login

- **Descrição:** Verifica se a janela é criada corretamente e se está visível para o utilizador.
- **Resultado esperado:** A janela de login deve estar visível para o utilizador.
- **Resultado obtido:** Teste bem sucedido, a janela de login foi mostrada ao utilizador como esperado.

5.4.3 Teste do método Main

- **Descrição:** Certifica que o método *main* do sistema executa corretamente.
- **Resultado esperado:** O método *main* deve ser executado sem erros.

- **Resultado obtido:** O teste foi aprovado com sucesso, o método é executado sem falhas.

Os testes realizados mostram que os procedimentos principais de inicialização do sistema estão funcionais. Os testes dão uma garantia de que o sistema pode ser inicializado com segurança e que as operações essenciais são corretamente configuradas, assim como a conexão e a interface gráfica.

5.5 Testes no médico

O objetivo destes testes unitários é validar a criação, manipulação e verificação de atributos e operações relativas à classe Medico. Estes testes garantem que os dados do médico, tais como os respetivos horários, sejam geridos corretamente.

5.5.1 Teste de criação de médico

- **Descrição:** Certifica que um objeto médico é criado de forma correta com os atributos esperados.
- **Resultado esperado:** O médico deve ser criado com atributos como o nome, número da ordem e especialidade corretamente atribuídos.
- **Resultado obtido:** O teste foi aprovado, o médico foi criado corretamente.

5.5.2 Teste de especialidade do médico

- **Descrição:** Verifica se a especialidade imposta ao médico está correta.
- **Resultado esperado:** Deve ser retornada a especialidade correta atribuída ao médico (método *getEspecialidade*).
- **Resultado obtido:** O teste foi um sucesso, a especialidade foi atribuída corretamente.

5.5.3 Teste do número da ordem do médico

- **Descrição:** Verifica se o número de ordem do médico é o correto.
- **Resultado esperado:** Deve ser retornado o número da ordem atribuído ao médico (método *getNumOrdem*).
- **Resultado obtido:** Teste aprovado, o número da ordem está correto.

5.5.4 Teste de criação de horários para o médico

- **Descrição:** Certifica que os horários criados para o médico são guardados corretamente.

- **Resultado esperado:** O objeto de horários do médico deve incluir a lista correta de horários atribuídos.
- **Resultado obtido:** Teste bem sucedido, os horários são criados e armazenados corretamente.

5.5.5 Teste de criação de vários horários para o médico

- **Descrição:** Verifica se são armazenados vários horários de forma correta.
- **Resultado esperado:** O objeto de horários do médico deve incluir todos os horários criados de forma correta, com os dados correspondentes de data e hora.
- **Resultado obtido:** Teste aprovado, múltiplos horários foram criados e armazenados corretamente.
- Os testes feitos em relação ao médico garantem que as operações estejam com o funcionamento correto.

5.6 Testes no paciente

Os seguintes testes têm como objetivo validar a criação e a manipulação de objetos da classe Paciente. Os testes garantem que os atributos principais, tais como o número de SNS, nome e contacto, são atribuídos e manipulados de forma correta.

5.6.1 Teste de criação de paciente

- **Descrição:** Certifica que um paciente é criado corretamente com os atributos disponibilizados.
- **Resultado esperado:** O paciente deve conter o número de SNS, nome e contacto corretos.
- **Resultado obtido:** Teste bem sucedido, o paciente foi criado corretamente.

5.6.2 Teste dos getters do paciente

- **Descrição:** Verifica se os métodos getters da classe Paciente retornam os valores esperados relativos à criação do paciente.
- **Resultado esperado:** Os métodos *getNumeroSNS*, *getNome*, *getContacto* devem retornar os valores corretos.
- **Resultado obtido:** Teste aprovado, os métodos getters funcionam de forma correta.

5.6.3 Teste de igualdade entre pacientes

- **Descrição:** Certifica que dois pacientes criados com o mesmos atributos são considerados iguais.

- **Resultado esperado:** Os dois pacientes não devem ser a mesma instância, porém os dados devem ser consistentes.
- **Resultado obtido:** Teste aprovado, os dados são iguais, mas as instâncias são distintas.

5.6.4 Teste de número SNS padrão

- **Descrição:** Verifica se o sistema guarda de forma correta o número SNS padrão de 9 dígitos.
- **Resultado esperado:** O número de SNS deve armazenar corretamente valores de 9 dígitos.
- **Resultado obtido:** O teste foi bem sucedido, o número de SNS foi guardado corretamente.

5.6.5 Teste de contato padrão

- **Descrição:** Certifica que o sistema armazena de forma correta valores que seguem o padrão esperado de 9 dígitos.
- **Resultado esperado:** O contacto deve armazenar corretamente valores com 9 dígitos.
- **Resultado obtido:** O teste foi aprovado, os valores armazenados para o contacto respeitam o padrão esperado.

Estes testes feitos em relação ao paciente garantem que a criação, manipulação e verificação de atributos do paciente funcionam corretamente.

5.7 Testes no registo clínico

Os testes em seguida têm como objetivo validar a criação e inicialização de registos clínicos. Esta classe é fundamental para armazenar e gerir o histórico médico de pacientes, que inclui doenças, alergias e operações realizadas.

5.7.1 Teste de criação de registo clínico

- **Descrição:** Verifica se o registo clínico é criado de forma correta com os atributos fornecidos, tais como o número de SNS, e verifica também se as listas de histórico de doenças, alergias e operações são inicializadas vazias.
- **Resultado esperado:** O número de SNS deve ser atribuído e todas as listas devem ser inicializadas como vazias.
- **Resultado obtido:** Teste aprovado, o registo clínico foi criado corretamente e as listas estavam vazias.

5.7.2 Teste de criação de registo clínico vazio

- **Descrição:** Certifica que o registo clínico é criado sem dados adicionais e se as listas são inicializadas vazias.
- **Resultado esperado:** Todas as listas do histórico de doenças, alergias e operações devem ser vazias.
- **Resultado obtido:** Teste aprovado, as listas foram inicializadas como esperado, vazias.

Estes testes realizados garantem que as entidades relacionadas com o registo clínico são criadas corretamente.

5.8 Testes na interação com a base de dados

O objetivo dos seguintes testes é validar os métodos responsáveis pela relação com os respetivos dados a pacientes e horários médicos. Esta classe pratica um papel importante na gestão de base de dados, garante a recuperação e modificação esperadas das afirmações armazenadas na base de dados.

5.8.1 Teste de criação de paciente

- **Descrição:** Verifica se o método *obterTodosPacientes* retorna uma lista de pacientes correta.
- **Resultado esperado:** A lista que será retornada deve ter pelo menos um paciente, com atributos como SNS, nome e contacto válidos.
- **Resultado obtido:** Teste aprovado, o método retorna uma lista correta de pacientes.

5.8.2 Teste de criação de paciente

- **Descrição:** Certifica que o método *criarPaciente* adiciona um novo paciente com os dados oferecidos.
- **Resultado esperado:** Este paciente deve ser adicionado corretamente na base de dados, pode ser validado posteriormente numa consulta.
- **Resultado obtido:** Teste bem sucedido, o paciente foi criado de forma correta.

5.8.3 Teste de obter todos os horários dos médicos

- **Descrição:** Verifica se o método *todosHorariosMedicos* retorna uma lista de horários com médicos ocupados.
- **Resultado esperado:** Esta lista deverá ter pelo menos um horário para cada médico, e os horários não poderão ser médicos.
- **Resultado obtido:** Teste aprovado, os horários dos médicos são recuperados corretamente.

Estes testes garantem que o sistema funciona corretamente em relação à comunicação entre o sistema e a base de dados.

5.9 Testes em dividir uma string

Os testes que se seguem têm como objetivo validar a funcionalidade do método responsável por dividir uma string em partes com base num delimitador funcional.

5.9.1 Teste com String válida

- **Descrição:** Certifica que o método divide de forma correta uma string que contém delimitadores numa lista de substrings.
- **Resultado esperado:** A lista a que deu resultado deve conter os elementos divididos de forma correta.
- **Resultado obtido:** Teste bem sucedido, a string foi dividida corretamente.

5.9.2 Teste com String vazia

- **Descrição:** Verifica como se comporta o método ao receber uma string vazia.
- **Resultado esperado:** A lista a que deu resultado deve conter apenas um elemento vazio.
- **Resultado obtido:** Teste aprovado, o método lida corretamente com strings vazias.

5.9.3 Teste com String nula

- **Descrição:** Assegura que o método dá erro ao receber uma string nula como entrada.
- **Resultado esperado:** Um erro deve ser lançado.
- **Resultado obtido:** Teste aprovado, confirma que o método aborda de forma correta esse cenário.

Estes testes garantem uma boa funcionalidade ao que diz respeito às strings, com a utilização devidamente correta dos delimitadores.

5.10 Testes em utilizadores

O objetivo destes testes é validar a criação e os métodos getters de objetos da classe Utilizador. Essa classe é essencial para representar os diferentes tipos de utilizadores do sistema, tais como médicos, funcionários e gestores.

5.10.1 Teste de criação de utilizar

- **Descrição:** Certifica que um utilizador é criado corretamente com os atributos fornecidos.
- **Resultado esperado:** O utilizador deve ter os valores corretos para ID, número de CC, nome, password e tipo de utilizador.
- **Resultado obtido:** Teste bem sucedido, o utilizador foi criado de forma correta.

5.10.2 Teste dos getters do utilizador

- **Descrição:** Verifica se os métodos getters da classe Utilizador retornam os valores esperados atribuídos durante a criação do objeto.
- **Resultado esperado:** Os métodos getters devem retornar os valores corretos.
- **Resultado obtido:** Teste aprovado, os métodos getters funcionam de forma correta.

Estes testes asseguram uma boa funcionalidade na gestão dos utilizadores, assim como a sua criação e o uso correto dos getters.

6 Anexos

Anexo 1: Caminho para o Manual de Uso - Manual de Uso da Aplicação
Caminho:Anexos\Manual De uso.pdf

Anexo 2: Caminho para os Testes - Testes feitos na aplicação
Caminho:Anexos\Testes.pdf

Anexo 3: Caminho para o Installer da aplicação
Caminho:Anexos\MediFlowInstallerScript.iss