1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

departamento de engenharia informática

MED SYNC

HOSPITAL MANAGEMENT SYSTEM

Unidade Curricular: Bases de Dados
Curso: Engenharia Informática
Data: 22/03/2024

ÍNDICE

INTRODUÇÃO

MEMBROS DA EQUIPA

Diogo José Gaspar Ferreira
Nº 2022220735
diogoferreira@student.dei.uc.pt

Luana Carolina Cunha Reis
Nº 2022220606
luanareis@student.dei.uc.pt

Rafael Teixeira da Silva
Nº 2019219259
rafaelsilva@student.dei.uc.pt

INTRODUÇÃO

DESCRIÇÃO DO PROJETO

No contexto de saúde atual, a precisão e eficiência na gestão hospitalar são de extrema importância para a garantia de um atendimento de qualidade aos pacientes, bem como o funcionamento adequado das instituições de saúde. Neste cenário, a implementação de um Sistema de Gestão Hospitalar (HMS) assume um papel crucial. O HMS é uma solução tecnológica que visa otimizar e simplificar processos operacionais, abrangendo desde a gestão de pacientes até a coordenação de recursos e faturamento.

O sistema ***MedSync*** propõe-se a responder a estas questões, desenvolvendo um sistema centrado num banco de dados capaz de atender às demandas complexas e dinâmicas de um ambiente hospitalar. Será projetado e implementado seguindo as melhores práticas da indústria de desenvolvimento de *software*, garantindo alta qualidade, desempenho e segurança.

Algumas das funcionalidades-chave propostas incluem...

1) Gestão de pacientes
2) Gestão de empregados
3) Agendamento de consultas e hospitalizações
4) Gestão de prescrições e medicamentos
5) Faturação e pagamentos
6) Autenticação e segurança
7) Relatórios e análises

DEFINIÇÕES

TRANSAÇÕES

Neste contexto, transações representam operações atómicas que devem ter êxito ou falhar como um todo.

Por exemplo, ao realizar uma alteração na base de dados, poderão estar envolvidas múltiplas operações de menor dimensão. Como tal, para garantir a integridade e consistência dos dados, este conjunto de ações deve ser tratado como uma única transação. Se alguma das operações falhar, toda a transação deverá ser revertida, garantindo que o banco de dados permanece num estado consistente.

Por exemplo...

- **Gerir pacientes/empregados:** inserir/remover uma pessoa da base de dados ou alterar as informações da mesma.
- **Coordenar consultas, hospitalizações e cirurgias:** abrange a criação/remoção/alteração das anteriores, tal como permite gerir o pessoal (pacientes/empregados) associado.
- **Gerir prescrições:** é possível criar receitas para medicamentos, alterar os medicamentos/dosagem de uma receita e cancelar receitas.
- **Gerir faturação:** permite gerar faturas, especificar os métodos de pagamento das mesmas e verificar se estas estão a pagas ou não. É possível alterar a informação de uma fatura após a sua criação.
- **Autenticação de utilizadores:** sistema de *login* que verifica se as credencias de acesso são válidas.

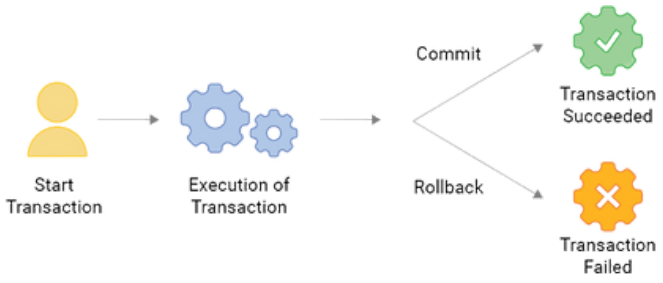

Fig. 1 - Funcionamento de transação

DEFINIÇÕES

CONFLITOS DE CONCORRÊNCIA

Os conflitos de concorrência podem surgir quando múltiplos utilizadores tentam aceder ou modificar os mesmos dados simultaneamente, resultando em inconsistências ou perda de dados se não forem geridos adequadamente.

Alguns exemplos:

- **Sobreposição de marcações:** Um possível conflito seria, por exemplo, uma enfermeira ter de estar presente em duas cirurgias em simultâneo ou um médico ter duas consultas agendadas para o mesmo horário. Este conflito é relativamente simples de controlar, recorrendo ao uso de *row-level locks* sobre os registos que já temos de uma dada pessoa (paciente ou empregado), garantindo que não são feitas marcações para o mesmo horário. Podemos até implementar um sistema que sugira um horário/médico/enfermeiro diferente.
- **Incompatibilidade de prescrições:** Outro conflito seria o facto de certos medicamentos não poderem ser tomados em simultâneo. Recorrendo outra vez ao uso de *row-level locks* podemos controlar as alterações de prescrições (deteção automática de conflitos invalidando as alterações feitas a uma prescrição).

Fig. 2 - Tratamento de conflitos

PLANO DE DESENVOLVIMENTO

TAREFAS PLANEADAS

Planeamento inicial

(01) Definir os objetivos e requisitos do projeto;
(02) Estabelecer prazos e metas;

Análise e design do banco de dados

(03) Levantamento dos requisitos de dados;
(04) Criar o modelo conceitual (modelo *Entity-Relationship*);
(05) Desenvolver o modelo físico (tabelas, colunas, chaves primárias/estrangeiras, ...);
(06) Projetar e implementar os endpoints da REST API;
(07) Integração dos mecanismos de autenticação e autorização de utilizadores na API;
(08) Testar a funcionalidade da API através do *Postman;*

Desenvolvimento das funcionalidades

(09) Implementar a lógica de negócios para as funcionalidades propostas;
(10) Garantir a integridade dos dados por meio de transações e controlos de concorrência;
(11) Realizar testes unitários e de integração para validar o correto funcionamento;

Testes e depuração

(12) Realizar testes abrangentes para identificar e corrigir erros;
(13) Depurar e otimizar o código conforme necessário, de forma a garantir desempenho e estabilidade.

Documentação e Relatório

(14) Documentar o processo de desenvolvimento, incluindo decisões de design, problemas e soluções implementadas.

PLANO DE DESENVOLVIMENTO

DIVISÃO INICIAL DE TRABALHO

No decorrer do desenvolvimento deste projeto, será adotada uma abordagem colaborativa, onde todos os membros terão igual oportunidade de contribuir de forma ativa. Embora seja possível dividir tarefas, a nossa prioridade será promover discussões conjuntas para garantir uma colaboração eficaz, efetiva e equitativa, valorizando a diversidade de perspectivas e experiências.

TIMELINE

Tarefas	15-21 mar	22-28 mar	29-04 abril	05-11 abril	12-18 abril	19-25 abril	26-02 maio	03-09 maio	10-16 maio	17-23 maio
(1)										
(2)										
(3)										
(4)										

Fig. 3 - Diagrama de Gantt

PLANO DE DESENVOLVIMENTO

TIMELINE

Tarefas	15-21 mar	22-28 mar	29-04 abril	05-11 abril	12-18 abril	19-25 abril	26-02 maio	03-09 maio	10-16 maio	17-23 maio
(05)										
(06)										
(07)										
(08)										
(09)										
(10)										
(11)										
(12)										
(13)										
(14)										

Fig. 3 - Diagrama de Gantt

MANUAL DE INSTALAÇÃO

SOFTWARE CRUCIAL:

- Postgres
- Python3 ou pip3
- Postman

BIBLIOTECAS NECESSÁRIAS (PYTHON):

- Flask (request, jsonify)
- flask_jwt_extended
- werkzeug.security
- Logging
- Psycopg2
- Datetime (datetime, timedelta)
- RE

Uma maneira simples de instalá-las é digitando «pip install [nome da biblioteca desejada]» num terminal Python.

CONFIGURAÇÃO:

A primeira coisa a fazer é criar a sua base de dados no pgAdmin e inserir as tabelas necessárias. Para isso, abra o arquivo onde está o script SQL para criar as tabelas na Ferramenta de Consulta do pgAdmin e clique no botão "Executar". Isso vai criar as tabelas e os triggers necessários. Depois de criar as tabelas e os dados, abra o IDE e execute o script Python "api_code.py" que acompanha o arquivo mencionado. Certifique-se de verificar a função "db_connection()" no código Python, porque ela deve ter as especificações corretas da base de dados para ficar conectada.

Finalmente, lance o Postman e execute o script "BD.postman_collection.json" que acompanha os outros arquivos. Com tudo configurado, pode agora consultar o Manual do Utilizador para obter todas as informações necessárias para começar a testar!

MANUAL DO UTILIZADOR

REGISTER

- Método: POST
- URL: http://localhost:8080/dbproj/register/<role>
- Existem 4 registers: Register Patient, Register Assistant, Register Patient e Register Patient.
- Em todos os register temos de especificar as seguintes informações no body: “username”, “password”, “name”, “mobile_number”, “birth_date”, “address” e “email”.
- Para os empregados (Assistentes, Enfermeiros e Médicos) temos de especificar as informações do contrato tais como: “salary”, “start_date”, “duration” e “end_date” (opcional).
- Para os Enfermeiros temos de especificar a sua posição (“position”)
- Para os Médicos temos de especificar “license_info” e “specializations_ids”.

AUTHENTICATION

- Método: PUT
- URL: http://localhost:8080/dbproj/user
- Existem 4 autenticadores: Patient Authentication, Assistant Authentication, Nurse Authentication e Doctor Authentication.
- O body é igual para todos tendo apenas de especificar : “username”, “password”.
- Os autenticadores diferentes servem para identificar mais facilmente qual usar.

SCHEDULE APPOINTMENT

- Método: POST
- URL: http://localhost:8080/dbproj/appointment
- Body: “doctor_id” e “date”

SEE APPOINTMENTS

- Método: GET
- URL: http://localhost:8080/dbproj/appointment/<patient_user_id>
- Body: não é preciso especificar nada

SCHEDULE SURGERY

- Método: POST
- URL: http://localhost:8080/dbproj/surgery
- Body: “patient_id”, “doctor”, “nurses” e “date”
- Caso queiramos adicionar a uma hospitalização já existente é só usar o URL: http://localhost:8080/dbproj/surgery/<hospitalization_id>

MANUAL DO UTILIZADOR

GET PRESCRIPTIONS

- Método: GET
- URL: http://localhost:8080/dbproj/prescriptions/<patient_id>
- Body: não é preciso especificar nada

ADD PRESCRIPTIONS

- Método: GET
- URL: http://localhost:8080/dbproj/prescription/
- Body: “type” , “event_id”, “validity”, “medicines”
- Medicines é uma lista em que cada posição tem “medicine”, “posology_dose” e “posology_frequency”

EXECUTE
EXECUTE PAYMENT

- Método: POST
- URL: http://localhost:8080/dbproj/bills/<bill_id>
- Temos apenas de especificar “amount” e “payment_method”

LIST TOP 3 PATIENTS

- Método: GET
- URL: http://localhost:8080/dbproj/top3
- Neste body só precisamos de autenticar o token de um assistente.

DAILY SUMMARY

- Método: GET
- URL: http://localhost:8080/dbproj/daily/{year-month-day}
- Neste body só precisamos de autenticar o token de um assistente.

MONTHLY REPORT

- (Método: GET
- URL: http://localhost:8080/dbproj/report
- Neste body só precisamos de autenticar o token de um assistente.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

O desenvolvimento do *MedSync* representa não apenas um desafio técnico, mas também uma oportunidade de aplicar conceitos teóricos num contexto prático. Ao longo deste projeto, são postas em prática uma série de etapas, desde a fase de análise inicial até à implementação e teste das funcionalidades.

Durante este processo, são consolidados e aprimorados conhecimentos sobre conceitos fundamentais de modelagem de dados, transações de banco de dados, segurança, gestão de concorrência, entre outros. Cada etapa exige a tomada de decisões cuidadosas, considerando não apenas os requisitos funcionais, mas também os aspectos técnicos e de usabilidade do sistema.

No final, o resultado é um Sistema de Gestão Hospitalar robusto e funcional, capaz de atender às necessidades dos profissionais de saúde e contribuir para uma boa prestação de serviços. Este projeto não só permitiu aplicar os conhecimentos adquiridos em contexto de aula como também nos proporcionou uma experiência valiosa de trabalho em equipa e desenvolvimento de sistemas complexos num ambiente do mundo real.

Fig. 7 - Imagem ilustrativa