Universidade de Coimbra

Departamento de Engenharia Informática

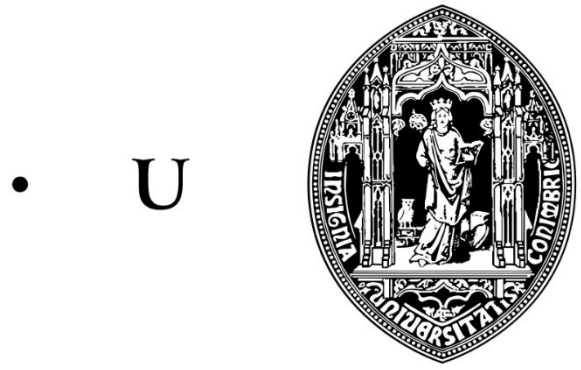
Base de dados

2017 - 2018

Relatório final

Autores

Gonçalo Oliveira Amaral 2015249122
Filipe Pinto Lopes 2011150922



• U C •

UNIVERSIDADE DE COIMBRA

Índice

Descrição do projeto

Tendo como ponto de partida o tema do projeto de sistema distribuídos ser o mesmo, decidimos melhorar a implementação anteriormente feita para a meta 1 de sistemas distribuídos, implementada em Java, conectando este sistema a um sistema de gestão de base de dados (SGBD). Para SGBD usamos Oracle Database Express Edition 11g Release 2, a usada nas aulas práticas. Todo o código fonte está presente nos pacotes .jar.

Descrição arquitetural

O sistema implementado divide-se em 5 unidades:

1. DataServer - Unidade que comunica diretamente com o SGBD através dos drivers JDBC, com o Server através de RMI e com a Console através de RMI.
2. Console - Consola de administração onde são efetuadas grande partes das operações CRUD. Unidade que comunica com o DataServer através de RMI.
3. Server - Servidor TCP, mesa de voto onde são feitas listagens e envia para o DataServer votos onde é apenas identificada a lista e a eleição que foi escolhida e logs de votos onde apenas são identificados a pessoa, a mesa voto e a eleição em que votou. Unidade que comunica com o DataServer através de RMI e com o Terminal através de TCP.
4. Terminal - Terminal de voto onde é feita a última interação com o utilizador, a escolha da lista em que o mesmo vota. Unidade que comunica com o Server através de TCP.
5. Oracle Database Express Edition 11g Release 2 - SGBD responsável por guardar os dados do sistema de forma persistente e tornas todas as operações CRUD mais eficientes e simples. Unidade que comunica com o DataServer.

Problemas da implementação

Apesar de a implementação ter em consideração os problemas que o tema expõe, como a autenticação de utilizadores, foram identificados alguns problemas que não foram solucionados. Primeiro, no caso de uma falha de conexão entre o Server e o DataServer, é garantido que não existem dois logs iguais mas como o log e o voto são enviados em separado, pode ocorrer duplicação de um voto. Segundo, existe redundância na tabela de logs, poderíamos obter a eleição através da mesa de voto. Esta redundância foi usada para simplificar certas queries feitas ao SGBD, não complicando a implementação. não

implementada a edição de uma pessoa. Terceiro, não foram tratados os casos em que a eliminação de uma entrada estava restringida (eliminar uma faculdade quando existe um departamento associado à mesma). Quarto, não foram incluídas operações que manipulam o estado da mesa de voto. Finalmente, termos de concorrência, a utilização de primary keys em todas as tabelas, o uso de sequências e o facto do método executeQuery do JDBC efetuar um commit sempre que é chamado impedem grande parte dos problemas de concorrência.

Manual de instalação

A aplicação requer a instalação do SGBD, Oracle Database Express Edition 11g Release 2, outros poderão funcionar desde que sejam da família Oracle DB apesar de não terem sido testadas. Deverá de seguida executar o ficheiro setup.sql para criar o utilizador, tabelas e sequências requeridas, e remover anteriores se existirem.

Manual de utilização

A aplicação é dividida em 4 pacotes JAR. Após todas as instruções de instalação serem cumpridas, deverá inicializar o SGBD, garantir que o ficheiro policy.all está na mesma pasta que o pacote a executar e que o pacote ojdbc14.jar está na mesma pasta que o pacote DataServer.jar.
Para usar:

1. A consola de administração deverá inicializar pelo menos uma instância do DataServer e pelo menos uma instância do Console.
2. A mesa de voto deverá inicializar pelo menos uma instância do DataServer, pelo menos uma instância do Server e pelo menos uma instância do Terminal.

A implementação dos menus é simples, por isso achamos que será intuitivo a utilização dos mesmos.

Descrição de Interface

Finalmente iremos descrever a interface remota que contém os métodos que irão resultar em queries ao SGBD.

1. Person
    a. int createPerson(Person person)
        i. Recebe uma pessoa,

        ii. Cira a pessoa
        iii. Devolve o CC da pessoa ou -1 se a pessoa já existir.
    b. ArrayList<Person> listStudentsFromDepartment(String departmentName)
        i. Recebe um departamento
        ii. Obtém todos os estudantes desse departamento
        iii. Devolve a lista de estudantes obtidos

    c. ArrayList<Person> listPeopleOfType(String type)
        i. Recebe um tipo de pessoa (Student - Estudante, Employee - Funcionário, Teacher - Professor)
        ii. Obtém todas as pessoas desse tipo
        iii. Devolve a lista de pessoas obtida
2. Faculty
    a. String createFaculty(String name)
        i. Recebe nome da faculdade
        ii. Cria nova faculdade
        iii. Devolve nome da faculdade ou null se a faculdade já existir
    b. void removeFaculty(String name)
        i. Recebe o nome da faculdade
        ii. Apaga a faculdade
    c. String updateFaculty(Faculty faculty, Faculty newFaculty)
        i. Recebe a faculdade e a faculdade atualizada
        ii. Atualiza a faculdade
        iii. Devolve o nome da faculdade atualizada ou null se a faculdade atualizada já existir
    d. ArrayList<Faculty> listFaculties()
        i. Obtém todas as faculdades
        ii. Devolve as faculdades obtidas
3. Department
    a. String createDepartment(Department department)
        i. Recebe um departamento
        ii. Cria departamento
        iii. Devolve o nome do departamento ou null se o departamento já existir
    b. void removeDepartment(String name)
        i. Recebe o nome do departamento

        ii. Remove departamento
    c. String updateDepartment(Department department, Department newDepartment)
        i. Recebe departamento e versão atualizada do departamento
        ii. Atualiza o departamento
        iii. Devolve o nome do departamento atualizado ou null se o departamento atualizado já existir
    d. ArrayList<Department> listDepartments(String facultyName)
        i. Recebe o nome de uma faculdade
        ii. Obtém os departamentos dessa faculdade
        iii. Devolve lista de departamentos obtida
4. Election
    a. int createElection(Election election)
        i. Recebe eleição
        ii. Cria eleição
        iii. Devolve id da eleição criada ou -1 se a eleição já existir
    b. Election getElection(int id)
        i. Recebe o id de uma eleição
        ii. Obtém eleição com o respectivo id
        iii. Devolve a eleição obtida ou null se não exitir
    c. ArrayList<Election> listElections(String type, String subtype)
        i. Recebe tipo (General - Geral, Nucleous - Núcleo) e subtipo (Student - Estudante, Employee - Funcionário, Teacher - Professor, no caso de a eleição ser do tipo General e o nome do departamento se o tipo da eleição for Nucleous)
        ii. Obtém eleições de acordo com esse tipo e subtipo
        iii. Devolve lista obtida
5. VotingList
    a. int createVotingList(VotingList votingList)
        i. Recebe lista
        ii. Cria lista
        iii. Devolve id da lista criada ou devolve -1 se a inserção da lista falhar
    b. void removeVotingList(int id)
        i. Recebe id de uma lista
        ii. Remove lista com o id

    c. ArrayList < VotingList > listVotingLists(int electionID)
        i. Recebe id de eleição
        ii. Obtém todas as listas de uma eleição
        iii. Devolve lista obtida
6. VotingListMember
    a. int addCandidate(int votingListID, int personCC)
        i. Recebe id de uma lista e o CC da pessoa
        ii. Associa pessoa com o CC à lista
        iii. Devolve o CC da pessoa ou -1 se essa pessoa já estiver na lista
    b. void removeCandidate(int votingListID, int personCC)
        i. Recebe id de uma lista e o CC de uma pessoa
        ii. Remove associação entre a lista e a pessoa
    c. ArrayList < Person > listCandidates(int votingListID)
        i. Recebe id de uma lista
        ii. Obtém todas as pessoas associadas a essa lista
        iii. Devolve a lista obtida
7. VotingTable
    a. int createVotingTable(VotingTable votingTable)
        i. Recebe mesa de voto
        ii. Cria mesa de voto
        iii. Devolve id da mesa de voto criada ou -1 se a mesa já existir
    b. void removeVotingTable(int id)
        i. Recebe o id de uma mesa de voto
        ii. Remove mesa de voto com o id
    c. ArrayList < VotingTable > listVotingTables(int electionID)
        i. Recebe id de eleição
        ii. Obtém todas as mesas de voto de uma eleição
        iii. Devolve lista obtida
    d. ArrayList<VotingTable> getVotingTables(String departmentName, int cc)
        i. Recebe nome de um departamento e um CC de uma pessoa
        ii. Obtém eleições em que a pessoa com o CC pode votar
        iii. Devolve lista obtida
8. VotingLog
    a. boolean sendLog(VotingLog log)
        i. Recebe log

    ii. Cria log
    iii. Devolve true se o log foi inserido com sucesso ou no caso oposto false

9. Vote
    a. void sendVote(int electionID, String votingList)
        i. Recebe id de uma eleição e o nome de uma lista
        ii. Incrementa número de votos para a eleição com o id
            1. Se a lista for Branco, incrementa o número de votos em branco da eleição com o id.
            2. Se a lista for Nulo, incrementa o número de votos nulos da eleição com o id.
            3. Caso contrário, procura lista e incrementa o número de votos dessa lista.
10. Credential
    a. Credential getCredentials(int cc)
        i. Recebe CC de uma pessoa
        ii. Obtém número e password da pessoa com o CC
        iii. Devolve as credenciais obtidas ou devolve null se a pessoa nao existir
11. Result
    a. ArrayList<Result> getResults(int electionID)
        i. Recebe id de uma eleição
        ii. Obtêm número de votos de cada lista (incluindo brancos e nulos)
        iii. Devolve lista obtida

Diagramas

Diagrama Entidade Relacionamento

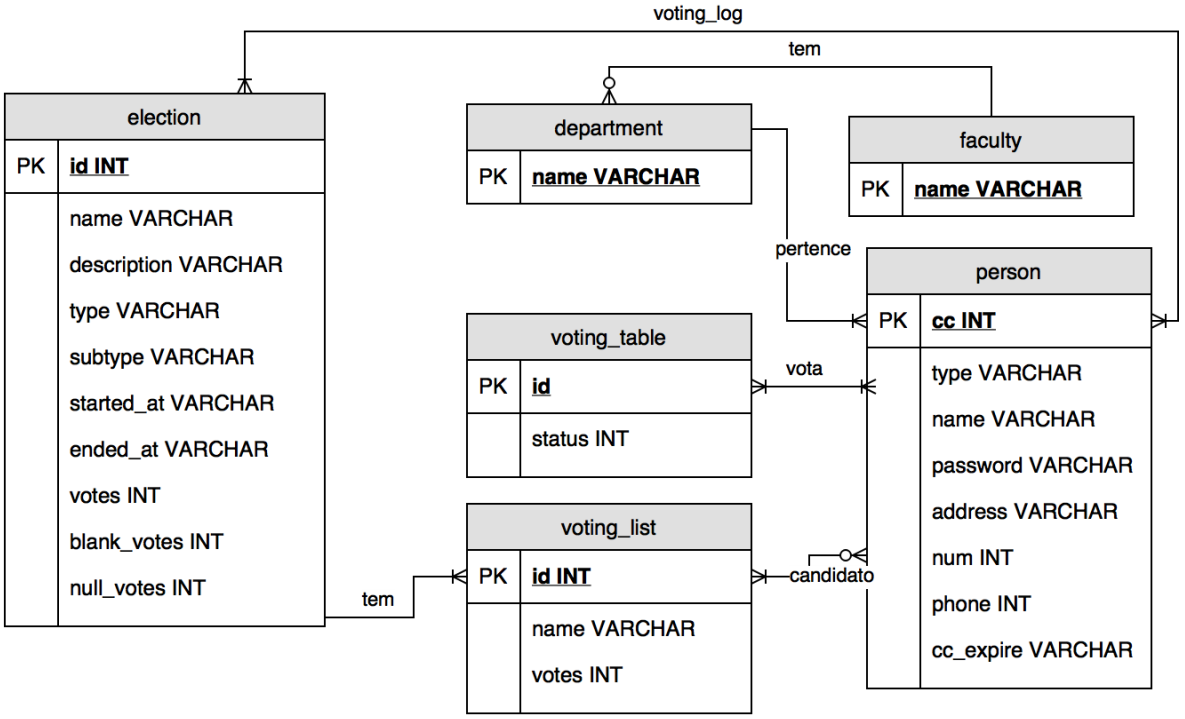


Diagrama de dados relacional