UNIVERSIDADE FEDERAL DE VIÇOSA
UFV - Campus Florestal

ARQUITETURA DE SOFTWARE

Gláucia Braga e Silva

Especificações das Classes

Álvaro Gomes Da Silva Neto - 5095
Danilo Matos De Oliveira - 5073
Jéssica Cristina Carvalho - 4686
Ítallo Winícios Ferreira Cardoso - 5101

Florestal - MG

2024

Sumário

Classe *ControladorProfessor*

Função: Gerenciar operações relacionadas aos professores, incluindo login, cadastro e listagem de turmas.

Métodos:

1. **fazerLogin(email: string, senha: string) : boolean**
    a. **Parâmetros**:
        i. `email`: O email do professor.
        ii. `senha`: A senha do professor.
    b. **Descrição**: Autentica o professor verificando suas credenciais diretamente no banco de dados. Retorna `true` se a autenticação for bem-sucedida, caso contrário, retorna `false`.

2. **cadastrarProfessor(nome: string, email: string, senha: string) : boolean**
    a. **Parâmetros**:
        i. `nome`: O nome do professor.
        ii. `email`: O email do professor.
        iii. `senha`: A senha para a conta do professor.
    b. **Descrição**: Cadastra um novo professor no sistema. Retorna `true` se o cadastro for bem-sucedido, caso contrário, retorna `false`.

3. **listarTurmas(idProfessor: int) : List<Turma>**
    a. **Parâmetros**:
        i. `idProfessor`: O identificador único do professor.
    b. **Descrição**: Retorna uma lista de turmas associadas ao professor especificado pelo `idProfessor`.

➔ **Orietações:**

- ◆ Garanta que a listagem de turmas retorne somente as turmas associadas ao professor autenticado.

➔ **Sugestões de Teste**:

- ◆ Testar o login com credenciais corretas e incorretas.
- ◆ Testar o cadastro de professores com dados válidos e inválidos.
- ◆ Verificar se a listagem de turmas retorna a lista correta para um professor específico.

Classe *ControladorTurma*

Função: Gerenciar operações relacionadas às turmas, incluindo listagem, remoção, cadastro e adição de alunos.

Métodos:

1. **cadastrarTurma(nomeTurma : string, idProfessor : int) : boolean**
   a. **Parâmetros**:
      i. `nomeTurma: O nome da turma.`
      ii. `idProfessor: O identificador único do professor`.
   b. **Descrição**: Cria uma nova turma associada a um professor. Retorna `true` se o cadastro for bem-sucedido, caso contrário, retorna `false`.
   c. Utilizando o idProfessor, deve-se verificar o cadastro do professor.

2. **vincularAlunoTurma(codigoTurma: int, aluno: ClasseAluno) : boolean**
   a. **Parâmetros**:
      i. `codigoTurma`: O identificador único da turma.
      ii. `nomeAluno`: O objeto `ClasseAluno` a ser adicionado.
   b. **Descrição**: Adiciona um aluno a uma turma especificada pelo `codigoTurma`. Retorna `true` se a adição for bem-sucedida, caso contrário, retorna `false`.
3. **removerTurma(codigoTurma: int) : boolean**

a. **Parâmetros**:
    i. `codigoTurma`: O identificador único da turma.
b. **Descrição**: Com base no *codigoTurma* passado como parâmetro a função remove a turma cadastrada e retorna `true` se a remoção for bem-sucedida, caso contrário, retorna `false`.

➔ **Orientações**:
- ◆ Assegure-se de que as turmas sejam corretamente associadas ao professor.
- ◆ Verifique que a remoção de turmas também remova todas as associações de alunos.

➔ **Sugestões de Teste**:
- ◆ Testar a remoção de uma turma e verificar a cascata de remoção.
- ◆ Verificar se o cadastro de turmas com dados válidos e inválidos é tratado corretamente.
- ◆ Testar a adição de alunos a uma turma específica e verificar a atualização na lista de alunos.

Classe *Professor*

Descrição: Representa os professores no sistema.

Atributos:

1. `idProfessor: int`: Identificador único do professor.
2. `nome: string`: Nome do professor.
3. `email: string`: Email do professor.
4. `senha: string`: Senha do professor.
5. `listaTurmas: List<Turma>`: Lista de turmas associadas ao professor.

Métodos: getters e setters públicos para os atributos listados acima.

Classe *Turma*

Descrição: Representa uma turma no sistema.

Atributos:

- `nomeTurma: string`: Nome da turma
- `codigoTurma: int`: Código da turma.
- `listaAlunos: List<Aluno>`: Lista de alunos pertencentes à turma.

➔ **Orientações**:

- ◆ O nomeTurma será formado pela série da turma e seu nome, por exemplo: 3° Ano B.
- ◆ Para o código da turma, ele é formado por 4 dígitos gerados aleatoriamente (não podem repetir), com o intuito de identificar a turma.

Métodos: quatidadeAlunos, método criado para consultar a quantidade de alunos

Getters e setters públicos para os atributos listados acima.

Classe *Aluno*

Descrição: Representa um aluno no sistema.

Atributos:

- `nomeAluno: string`: Nome do aluno.
- `score: int`: Score do aluno, faz referência a pontuação do aluno no decorrer do jogo
- `codigoTurma: int`: Referência à turma à qual o aluno pertence.

Métodos: getters e setters públicos para os atributos listados acima.