**PROJETOS IV BIMESTRE – PROGRAMAÇÃO AVANÇADA PARA WEB**

**Diagrama

Descrição gerada automaticamente**

Construa uma aplicação web composta de front-end e back end.

O front-end deve ser escrito em Javascript.

O back-end pode ser dividido em duas grandes partes, a área do professor e a área do aluno.

A área do aluno deve ser escrita em typeScript.

A área do professor do professor deve ser escrita em Javascript;

*O Código fonte da aplicação deve ser entregue no class room.*

*Pelo tamanho da aplicação é impossível que sejam apresentados projetos muito semelhantes, quando identificada cola, todos os alunos envolvidos serão punidos. Não haverá distinção entre alunos que colaram e alunos que passaram cola.*

***Mantenha no banco de dados registros para facilitar o processo de avaliação.***

***Dica: Faça o projeto de roteamento e divida a tarefa entre os elementos do grupo.***

***Exemplo de projeto de roteamento.***

|  |  |  |  |
| --- | --- | --- | --- |
| *rota* | *verbo* | *Padrão envio* | *Padrão resposta* |
| */turmas* | *get* | */turmas/id* | { “status”: true,  “msg”:”'executado com sucesso”,  “código”: “'201”,  dados: [ {“idTurma”:1,” nome”:”1AUNA”,” abreviacao”:”1A”,”ano”:2023 } ] } |

Itens do projeto.

1. Para cada tabela no banco de dados crie um CRUD completo no padrão Rest.
   1. Area do professor: Javascript
      1. CRUD professores
      2. CRUD Disciplinas
      3. CRUD Turmas
      4. CRUD Alunos
      5. CRUD Notas
      6. CRUD pedido de revisão
      7. CRUD histórico de alterações
   2. Área do aluno: TypeScript
      1. Classe ver nota
      2. Classe pedir revisão
      3. Classe histórico de alterações.
2. Construa um font end em Javascript, html e css para tornar a aplicação viável.
   1. Utilize como base o material de aula para fazer a interface front-end do CRUD, com filtros etc.
   2. A navegação da aplicação deve ser totalmente funcional com botões de voltar, e links para acessar todos os recursos.
3. Area do professor:
   1. Login do professor (JWT).
   2. Só o professor do tipo administrador pode cadastrar professores, Disciplinas, turmas e alunos.
   3. Lançar nota. Permitir que o professor lance apenas a nota dele, mas ele pode visualizar as notas dos outros professores. Utilize tabelas. Apresente também na tabela a média se a nota for de projetos.
   4. Caso o lançamento da nota seja de provas apresente também a nota que foi lançada pelos outros professores. Utilize uma tabela.
   5. No lançamento de notas permita a possibilidade de filtrar por nome, ou por matrícula.
   6. Apresentar a média, da turma, mediana e desvio padrão.
4. Área do aluno
   1. Login de aluno (JWT).
   2. Lista de disciplinas
   3. Ver notas dele. Apresente em forma de tabela. Mostre também a média da turma na disciplina. Apresente a nota em vermelho se estiver abaixo da média da sala na disciplina.
   4. Pedir revisão
5. Na área do professor permita
   1. Criar um gráfico com um histograma de notas. O gráfico deve ser gerado por turma, ou seja, o professor selecione a turma e o gráfico é apresentado.
   2. exportar um arquivo csv com as notas da disciplina. O Professor pode escolher o bimestre correspondente ou todos e o tipo da nota.

|  |
| --- |
| -- MySQL Workbench Forward Engineering  SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;  SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;  SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';  -- -----------------------------------------------------  -- Schema colegiosUnivap  -- -----------------------------------------------------  -- -----------------------------------------------------  -- Schema colegiosUnivap  -- -----------------------------------------------------  CREATE SCHEMA IF NOT EXISTS `colegiosUnivap` DEFAULT CHARACTER SET utf8 ;  USE `colegiosUnivap` ;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`Aluno`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`Aluno` (  `matricula` INT UNSIGNED NOT NULL,  `nome` VARCHAR(128) NULL,  `email` VARCHAR(45) NULL,  `wpp` BIGINT NULL,  `senha` VARCHAR(32) NULL,  PRIMARY KEY (`matricula`),  UNIQUE INDEX `matricula\_UNIQUE` (`matricula` ASC))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`Professor`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`Professor` (  `registro` INT UNSIGNED NOT NULL,  `nome` VARCHAR(128) NULL,  `email` VARCHAR(45) NULL,  `senha` VARCHAR(32) NULL,  `tipo` INT NULL COMMENT '1 - Professor\n2 - Professor Administrador\n',  PRIMARY KEY (`registro`),  UNIQUE INDEX `registro\_UNIQUE` (`registro` ASC))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`Turma`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`Turma` (  `idTurma` INT UNSIGNED NOT NULL AUTO\_INCREMENT,  `nome` VARCHAR(45) NULL,  `abreviacao` VARCHAR(45) NULL,  `ano` INT NULL,  PRIMARY KEY (`idTurma`),  UNIQUE INDEX `idTurma\_UNIQUE` (`idTurma` ASC))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`Disciplina`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`Disciplina` (  `idDisciplina` INT UNSIGNED NOT NULL AUTO\_INCREMENT,  `nome` VARCHAR(45) NULL,  `Professor\_registro` INT UNSIGNED NOT NULL,  `Turma\_idTurma` INT UNSIGNED NOT NULL,  PRIMARY KEY (`idDisciplina`),  UNIQUE INDEX `idDisciplina\_UNIQUE` (`idDisciplina` ASC),  INDEX `fk\_Disciplina\_Professor1\_idx` (`Professor\_registro` ASC),  INDEX `fk\_Disciplina\_Turma1\_idx` (`Turma\_idTurma` ASC),  CONSTRAINT `fk\_Disciplina\_Professor1`  FOREIGN KEY (`Professor\_registro`)  REFERENCES `colegiosUnivap`.`Professor` (`registro`)  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `fk\_Disciplina\_Turma1`  FOREIGN KEY (`Turma\_idTurma`)  REFERENCES `colegiosUnivap`.`Turma` (`idTurma`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`Nota`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`Nota` (  `idNota` INT UNSIGNED NOT NULL AUTO\_INCREMENT,  `Disciplina\_idDisciplina` INT UNSIGNED NOT NULL,  `Aluno\_matricula` INT UNSIGNED NOT NULL,  `bimestre` INT NULL,  `nota` FLOAT NULL,  `ultimaAlteracao` DATETIME NULL,  `tipoNota` INT NULL COMMENT '1 - Projeto.\n2 - Prova.',  `fezLista` INT NULL,  PRIMARY KEY (`idNota`),  UNIQUE INDEX `idNota\_UNIQUE` (`idNota` ASC),  INDEX `fk\_Nota\_Disciplina1\_idx` (`Disciplina\_idDisciplina` ASC),  INDEX `fk\_Nota\_Aluno1\_idx` (`Aluno\_matricula` ASC),  CONSTRAINT `fk\_Nota\_Disciplina1`  FOREIGN KEY (`Disciplina\_idDisciplina`)  REFERENCES `colegiosUnivap`.`Disciplina` (`idDisciplina`)  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `fk\_Nota\_Aluno1`  FOREIGN KEY (`Aluno\_matricula`)  REFERENCES `colegiosUnivap`.`Aluno` (`matricula`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`PedidoRevisao`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`PedidoRevisao` (  `idPedidoRevisao` INT UNSIGNED NOT NULL AUTO\_INCREMENT,  `Nota\_idNota` INT UNSIGNED NOT NULL,  `descricao` VARCHAR(512) NULL,  `status` INT NULL COMMENT '1 - Não avaliado\n2 - Aceito e Alterado\n3 - Recusado e não alterado',  PRIMARY KEY (`idPedidoRevisao`),  UNIQUE INDEX `idPedidoRevisao\_UNIQUE` (`idPedidoRevisao` ASC),  INDEX `fk\_PedidoRevisao\_Nota1\_idx` (`Nota\_idNota` ASC),  CONSTRAINT `fk\_PedidoRevisao\_Nota1`  FOREIGN KEY (`Nota\_idNota`)  REFERENCES `colegiosUnivap`.`Nota` (`idNota`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `colegiosUnivap`.`HistoricoAlteracoes`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `colegiosUnivap`.`HistoricoAlteracoes` (  `Nota\_idNota` INT UNSIGNED NOT NULL,  `nota` FLOAT NULL,  `ultimaAlteracao` DATETIME NULL,  INDEX `fk\_Historico\_Nota1\_idx` (`Nota\_idNota` ASC),  CONSTRAINT `fk\_Historico\_Nota1`  FOREIGN KEY (`Nota\_idNota`)  REFERENCES `colegiosUnivap`.`Nota` (`idNota`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  SET SQL\_MODE=@OLD\_SQL\_MODE;  SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;  SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS; |