****

THIAGO DE OLIVEIRA MARTINS

**BELÉM**

**2021**

****

**SOFTWARE PARA VENDA DE PASSAGEM**

Trabalho elaborado pelo aluno: THIAGO DE OLIVEIRA MARTINS, UP20214389, apresentado ao curso de Análise e Desenvolvimento de Sistemas, da Universidade Paulista (UNIP), referente ao 4° semestre do curso de nível tecnológico superior, sob a orientação do professor: Rodney Victor.

**BELÉM**

**2021**

SUMÁRIO

**RESUMO**......................................................................................................................**4**

**ABSTRACT**..................................................................................................................**5**

**INTRODUÇÃO**.............................................................................................................**6**

**DESENVOLVIMENTO**.................................................................................................**7**

**CONCLUSÃO**............................................................................................................**18**

**REFERÊNCIAS**..........................................................................................................**19**

**RESUMO**

Atendendo aos requisitos da universidade com relação ao projeto integrador multidisciplinar (PIM VIII). Este projeto possui a parte prática e teórica e tem como principal objetivo o desenvolvimento de um programa para organização de tarefas acadêmicas, onde será cadastrado provas e eventos como se fosse uma agenda.

Palavras-chave: PIM, VIII, software, Tarefas, Nodejs, React.

**ABSTRACT**

Meeting the university's requirements regarding the multidisciplinary integrative project (PIM VIII). This project has a practical and theoretical part and its main objective is to develop a program for organizing academic tasks, where tests and events will be registered as if it were an agenda.

Keywords: PIM, VIII, software, Tasks, Nodejs, React.

**INTRODUÇÃO**

O software é uma ferramenta, que cada vez mais, traz novas funcionalidades que visam facilitar a execução de tarefas do ser humano. Visando essa funcionalidade foi criado um sistema de tarefas acadêmicas, para facilitar a organização acadêmica de alunos.

O sistema se consiste em uma aplicação web, onde será possível cadastrar, deletar, consultar e atualizar dados. Utilizando a linguagem Javascript e o MySQL, para realizar a criação do sistema.

Como forma de criar um projeto mais seguro e com alto desempenho foi utilizado o modelo MVC no sistema, com um detalhe, onde o back-and e front-end foi separado em duas aplicações diferentes. Onde falaremos mais sobre isso mais tarde.

Além disso o projeto tem como objeto fornecer já uma fácil utilização, sendo simples e objetivo, para que possa ser de fácil aprendizagem.

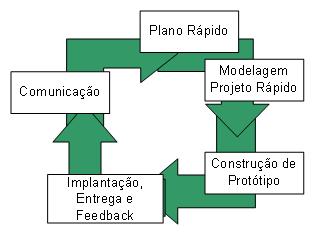
**DESENVOLVIMENTO**

**1 - Ciclo de vida**

O ciclo de vida usado foi o Codifica-Remenda, que tem como principal característica a codificação imediata, podendo com isso diminuir o tempo de entrega do projeto, por permitir a correção dos erros à medida que eles surgem.

Esse ciclo de vida traz com sigo alguns riscos, por não seguir nem um tipo de processo, o tornando de alto risco, para isso à outro ciclo que foi utilizado como ferramenta para auxiliar o Codifica-Remenda, o de Prototipagem.

A ferramenta de prototipagem, é um exemplar do que foi entendido dos requisitos do cliente, ele é considerado um ciclo de vida, mas no projeto foi usado apenas como uma ferramenta, função utilizada em outros ciclos de vida. Segue cinco fases: Plano Rápido; Modelagem de Projeto Rápido; Construção de protótipo; Implementação, Entrega e Feedback; Comunicação. A comunicação entre desenvolvedor e cliente é muito importante, pois ela vai ser necessária para corrigir erros, e melhorar o projeto.



O modelo de prototipagem (Pressman, adaptado)  
 **Fonte:** <https://www.devmedia.com.br/ciclos-de-vida-do-software/21099>

**2 – Ferramentas de Codificação**

**Javascript**

Javascript (frequentemente abreviado como JS) é uma linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma (protótipos, orientado a objeto, imperativo e, funcional). Juntamente com HTML e CSS, o Javascript é uma das três principais tecnologias da World Wide Web. Javascript permite páginas da Web interativas e, portanto, é uma parte essencial dos aplicativos da web. A grande maioria dos sites usa, e todos os principais navegadores têm um mecanismo Javascript dedicado para executá-lo.

* Sobre ECMAScript

ECMAScript é uma linguagem de programação, baseada em scripts, padronizada pela Ecma International na especificação ECMA-262 e ISO/IEC 16262. Esta linguagem é usada em tecnologias para Internet para a criação de scripts executados no cliente e no servidor (Node.JS), sendo inspirada na linguagem de programação Javascript, além permitir outras implementações independentes.

//modulos

const express = require('express')

const cors = require('cors')

const bodyParser = require('body-parser');

//config

const port = 5000;

const router = require('./router')

    //express

    const server = express()

    //cors

    server.use(cors())

    //bodyparse

    server.use(bodyParser.urlencoded({extended: true}))

    server.use(bodyParser.json())

server.use('/sa', router)

server.listen(port, '0.0.0.0', () => {

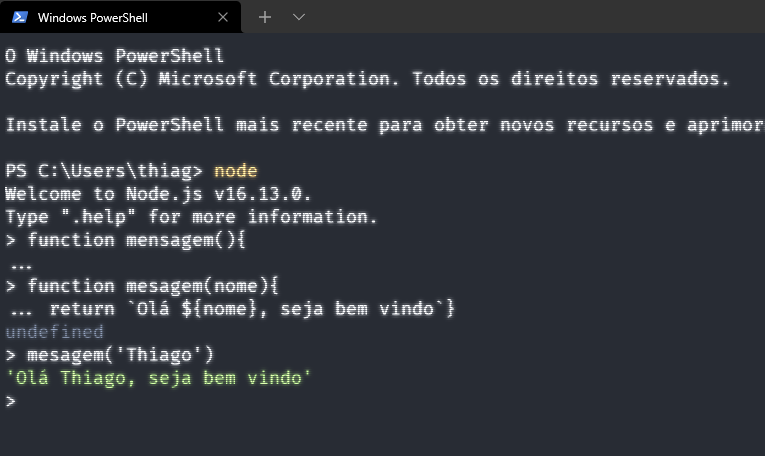
    console.log(`Server Run... \nhttp://localhost:${port}`)

})

O código acima apresenta um script escrito em linguagem ECMAScript, que inicia um servidor express.

**Node.js**

Node.js é um software de código aberto, multiplataforma, baseado no interpretador V8 do Google e que permite a execução de códigos JavaScript fora de um navegador web.

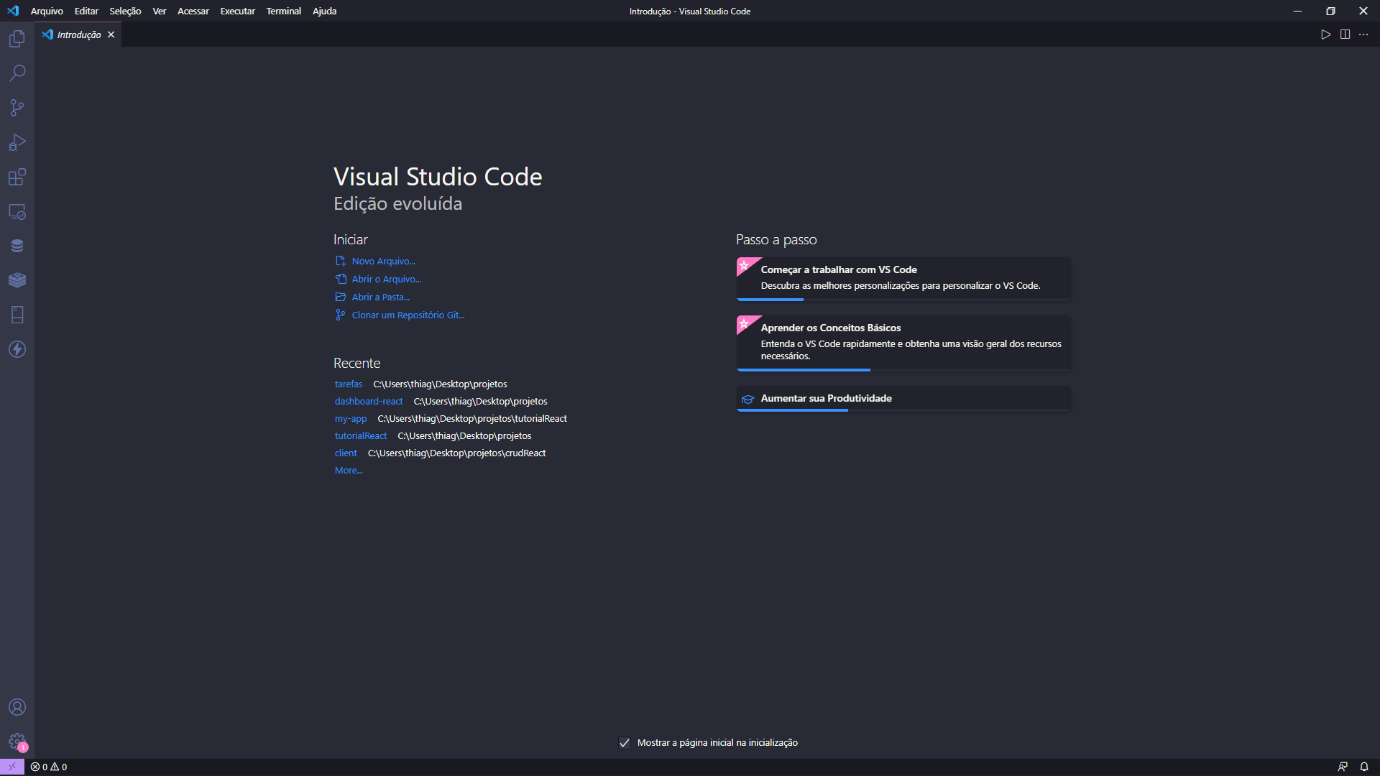
O runtime de Javascript é constituído pelos seguintes comandos: node package manager (npm), e npx (node package extractor); onde o primeiro tem o propósito de executar código armazenado num package de nodejs, instalando o software globalmente ou localmente, já o segundo tem o propósito de instalar a nível local o código instalado globalmente. Um exemplo disso é o npx create-react-app que tem como propósito instalar a nível local um template vazio de um site de react, pronto a ser usado, através de uma fórmula instalada com npm.

V8 do Nodejs

**React**

O React (também denominado React.js ou ReactJS) é uma biblioteca JavaScript de código aberto com foco em criar interfaces de usuário (frontend) em páginas web. É mantido pelo Facebook, Instagram, outras empresas e uma comunidade de desenvolvedores individuais. É utilizado nos sites da Netflix, Imgur, Feedly, Airbnb, SeatGeek, HelloSign, Walmart e outros.

**Visual Studio CODE**

O Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Ele inclui suporte para depuração, controle Git incorporado, realce de sintaxe, complementação inteligente de código, snippets e refatoração de código. Ele também é customizável, fazendo com que os usuários possam mudar o tema do editor, teclas de atalho e preferências. Ele é um software livre e de código aberto, apesar do download oficial estar sob uma licença proprietária.

Tela inicial do VSCODE

**3 – Requisitos**

O sistema deve possuir um CRUD (Create, Read, Update, Delete), que possibilite as quatro operações de banco de dados, sendo elas a inserção de dados, Leitura de dados, Atualização de dados e o Delete.

Deve ser mostrada na tela os resultados em uma tabela, para poder e um alerta para quando for feira uma operação no banco de dados.

Deve respeitar o relacionamento dos dados, e realizar tratamentos de erros para não fazer com que o sistema ficar travado.

**4 – Aplicação**

**4.1 Back-End**

O back-end é o responsável pelo tratamento de dados, onde ele fica do lado do servidor, sem a interferência do lado do cliente, ele é responsável por realizar as consultas do banco de dados e enviar para o cliente esses dados de forma segura.

Para o desenvolvimento do back-end foi utilizado o Nodejs que é uma ferramenta de código aberto que permite rodar código Javascript no lado do servidor.

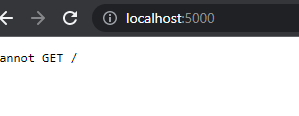
O nodejs vem com um conjunto de bibliotecas, funcionando como um framework com varias aplicações prontas para inserir no projeto. Dentro delas foram utilizadas algumas bibliotecas para realizar a construção do back-end.

1. Express: Express.js é um framework para Node.js que fornece recursos mínimos para construção de servidores web.
2. Cors: CORS é um pacote node.js para fornecer um middleware Connect / Express que pode ser usado para habilitar CORS com várias opções.
3. Body-parser: Analise os corpos das solicitações recebidas em um middleware antes de seus manipuladores, disponíveis na req.body.propriedade.
4. Mysql: Utilizado para comunicar com o banco de dados mysql – Falaremos mais sobre mysql a frente.
5. Nodemon: Nodemon é uma ferramenta que ajuda a desenvolver aplicativos baseados em node.js reiniciando automaticamente o aplicativo de nó quando mudanças de arquivo no diretório são detectadas.

O projeto seguiu uma estrutura baseado no MVC onde separa os Model, Control e View, onde o view fica em uma aplicação separada, como falaremos mais à frente.

Foi criado primeiramente os arquivos de configuração, como o server.js, router.js e bd.js.

O server.js é responsável por starter o servidor na porta 5000, onde poderemos buscar ele através do domínio ‘http://localhost:5000’, como mostra a imagem.



http://localhost:5000

Já nesse arquivo configuramos o cors para evitar erro de cors quando forem ser abertas em outro domínio e body-parser.

//modulos

const express = require('express')

const cors = require('cors')

const bodyParser = require('body-parser');

//config

const port = 5000;

const router = require('./router')

const server = express()

server.use(cors())

server.use(bodyParser.urlencoded({extended: true}))

server.use(bodyParser.json())

server.use('/sa', router)

O arquivo bd.js é responsável por fazer a conexão com o banco de dados, lá estão configurados o host, usuário, senha e o banco de dado que vai ser utilizado.

const mysql = require('mysql')

const bd = {

host: 'localhost',

user: 'saTarefas',

passwd: '\_8280',

bd: 'db\_tarefas\_academicas'

}

const connection = mysql.createConnection({

host: bd.host,

user: bd.user,

password: bd.passwd,

database: bd.bd,

dateStrings: true

})

connection.connect((erro) => {

if(erro) throw `Erro ao conectar com banco de dados : ${erro}`

console.log(`Conectado ao banco de dados: ${bd.bd}`)

})

module.exports = connection

O arquivo router.js é responsável por gerenciar as rotas do a API, nele será configurada a rota que deve ser acessado, sendo elas get, post, put, delete, onde get será utilizada para leitura de dados, post para inserção, put para atualização e delete para excluir dados.

Além de informar qual o tipo de rota, ele indica o que deve ser realizado após a rota ser chamada, nesse caso ele invoca uma função que está dentro do model do sistema, que será tratada mais a frente.

const express = require('express')

const router = express.Router()

//controllers

const professorController = require('./src/controllers/professorController')

const alunoController = require('./src/controllers/alunoController')

const turmaController = require('./src/controllers/turmaController')

const disciplinaController = require('./src/controllers/disiciplinaController')

const gradeController = require('./src/controllers/gradeController')

const provaController = require('./src/controllers/provaController')

const eventoController = require('./src/controllers/eventoController')

//CRUD Professor

router.get('/professores', professorController.buscarTodos)

router.get('/professor/:codigo', professorController.buscarUm)

router.post('/professor', professorController.addProfessor)

router.put('/professor/:codigo', professorController.alterProfessor)

router.delete('/professor/:codigo', professorController.delete)

//CRUD Turma

router.get('/turmas', turmaController.buscarTodos)

router.get('/turma/:id', turmaController.buscarUm)

router.post('/turma', turmaController.addTurma)

router.put('/turma/:id', turmaController.alterTurma)

router.delete('/turma/:id', turmaController.delete)

//grade-turma

router.get('/grade/:idTurma', gradeController.buscarUm)

router.post('/grade', gradeController.addGrade)

router.put('/grade/:turma/:dia', gradeController.alterGrade)

router.delete('/grade/:turma/:dia?', gradeController.delete)

//CRUD Aluno

router.get('/alunos', alunoController.buscarTodos)

router.get('/aluno/:matricula', alunoController.buscarUm)

router.post('/aluno', alunoController.addAluno)

router.put('/aluno/:matricula', alunoController.alterAluno)

router.delete('/aluno/:matricula', alunoController.delete)

//CRUD disciplina

router.get('/disciplinas', disciplinaController.buscarTodos)

router.get('/disciplina/:cod', disciplinaController.buscarUm)

router.post('/disciplina', disciplinaController.addDisciplina)

router.put('/disciplina/:cod', disciplinaController.alterDisciplina)

router.delete('/disciplina/:cod', disciplinaController.delete)

//Provas

router.get('/provas/:turma', provaController.buscarPorTurma)

router.get('/prova/:codigo', provaController.buscarPorProva)

router.post('/prova', provaController.addProva)

router.delete('/prova/:turma', provaController.delete)

router.put('/prova/:turma', provaController.alterProva)

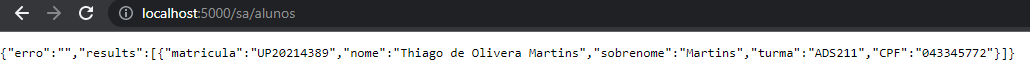
//eventos

router.get('/eventos/:turma?', eventoController.buscar)

router.post('/evento', eventoController.addEvento)

module.exports = router

As rotas são invocadas pela barra de navegação e devem ou enviadas através de um formulário do front-end. Caso não encontre a rota será informada como não sendo criada.



<http://localhost:5000/sa/alunos> retorna um Json com os dados

O controller dos dados foi criado para gerenciar o que cada rota invocada de fazer, ele se comunica com o Model (que vamos chamar de Service partir de agora), que se comunica com o banco de dados fazendo requisições.

Foi criado um controller para cada elemento da nossa aplicação, mas todos seguem o mesmo padrão, mudando apenas as características que são necessárias para sua funcionalidade.

const turmaService = require('../service/turmaService')

function formatarData(date){

let data = date.split('-')

let objData = {

ano: data[0],

mes: data[1],

dia: data[2]

}

return objData

}

module.exports = {

buscarTodos: async(req, res) => {

let json = {erro: '', results: [], curso: []}

try{

let turmas = await turmaService.buscarTodos()

let curso = await turmaService.curso()

for(i in curso){

json.curso.push({

codigo: curso[i].cod\_curso,

descricao: curso[i].nome\_curso,

periodos: curso[i].qtd\_periodo

})

}

for(i in turmas){

const data = formatarData(turmas[i].data\_inicio)

json.results.push({

turma: turmas[i].id\_turma,

sala: turmas[i].sala,

curso: turmas[i].curso,

periodo: turmas[i].periodo,

data: `${data.dia}/${data.mes}/${data.ano}`

})

}

}catch(e){

json.erro = `Erro de query: ${e}`

console.log(JSON.parse(json.erro))

}

res.json(json)

},

buscarUm: async(req, res) => {

let json = {erro: '', results: {}}

let idTurma = req.params.id

try{

let turma = await turmaService.buscarUm(idTurma)

if(turma){

json.results = turma

}else{

json.erro = 'Dados não retornados'}

}catch(e){

json.erro = `Erro de query: ${e}`

console.log(JSON.parse(json.erro))

}

res.json(json)

},

addTurma: async(req, res) => {

let json = {erro: '', results: {}}

let idTurma = req.body.turma

let periodo = req.body.periodo

let curso = req.body.curso

let data = req.body.data

let sala = req.body.sala

try{

if(idTurma && sala && curso && periodo){

let turma = await turmaService.addTurma(idTurma, periodo, curso, data, sala)

if(turma.affectedRows == 1)

json.results = `Turma ${idTurma} cadastrada`

}else{

json.erro = 'Campos não enviados'

}

}catch(e){

json.erro = `Erro de query: ${e}`

console.log(JSON.parse(json.erro))

}

res.json(json)

},

alterTurma: async(req, res) => {

let json = {erro: '', results: {}}

let idTurma = req.params.id

let sala = req.body.sala

let curso = req.body.curso

let periodo = req.body.periodo

try{

if(sala && curso && periodo){

let turma = await turmaService.alterTurma(idTurma, sala, curso, periodo)

json.results = turma

}else{

json.erro = 'Campos não enviados'

}

}catch(e){

json.erro = `Erro de query: ${e}`

console.log(JSON.parse(json.erro))

}

res.json(json)

},

delete: async(req, res) => {

let json = {erro: '', results: {}}

let idTurma = req.params.id

try{

let turma = await turmaService.delete(idTurma)

if(turma){

json.results = turma

}else{

json.erro = 'Nem um dado alterado'

}

}catch(e){

json.erro = `Erro de query: ${e}`

console.log(JSON.parse(json.erro))

}

res.json(json)

}

}

O código acima cria as funções que vão ser invocadas pelas rotas, toda a estrutura do código é pensada para ser feita de forma a garantir o desempenho do sistema, por esse motivo foi utilizada o async/await que nos dá a possibilidade criar funções assíncronas e manipular Promisses.

Além disse o padrão middleware foi utilizado, onde é feira a requisição de algo pelo corpo da requisição da rota, e enviado uma response, com os dados no formato Json (JavaScript Object Notation).

O servisse por sua vez faz a requisição no banco de dados das informações solicitadas pelo controller.

const db = require('../db')

module.exports = {

curso: () => new Promise((resolve, reject) => {

db.query('select \* from curso', (erro, result) => {

if(erro) reject(erro)

resolve(result)

})

}),

buscarTodos: () => new Promise((resolve, reject) => {

db.query(`select turma.id\_turma, turma.periodo, curso.nome\_curso as curso, turma.data\_inicio

from turma

inner join curso on turma.fk\_curso = curso.cod\_curso;`, (erro, results) => {

if(erro) reject(erro)

resolve(results)

})

}),

buscarUm: idTurma => new Promise((resolve, reject) => {

db.query('select \* from turma where id\_turma = ?;', [idTurma], (erro, results) => {

if(erro) reject(erro)

resolve(results)

})

}),

addTurma: (idTurma, periodo, curso, data, sala) => new Promise((resolve, reject) => {

db.query(`insert into turma(id\_turma, periodo, fk\_curso, data\_incio, sala)

values(?, ?, ?, ?, ?)`, [idTurma, periodo, curso, data, sala], (erro, results) => {

if(erro) reject(erro)

resolve(results)

})

}),

alterTurma: (idTurma, sala, curso, periodo) => new Promise((resolve, reject) => {

db.query(`update turma set

sala = ?, curso = ?, periodo = ?

where id\_turma = ?;`, [sala, curso, periodo, idTurma], (erro, results) => {

if(erro) reject(erro)

resolve(results)

})

}),

delete: idTurma => new Promise((resolve, reject) => {

db.query(`delete from turma where id\_turma = ?`, [idTurma], (erro, results) => {

if(erro) reject(erro)

resolve(results)

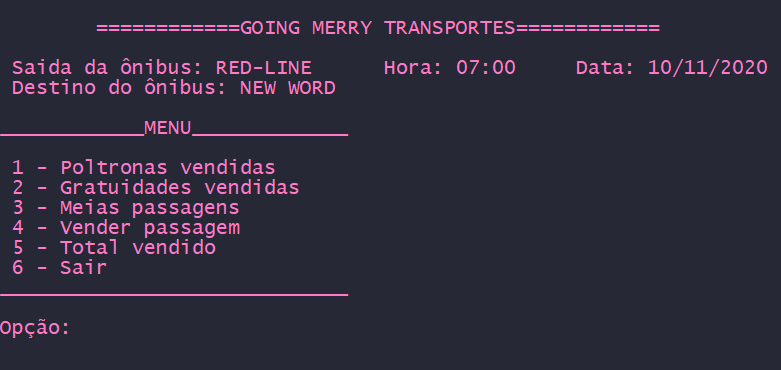
})

})

}

As opções possíveis são as seguintes, Poltronas Vendidas, Gratuidades Vendidas, Meias Passagens, Vender Passagem, Total Vendido, Sair, numeradas respectivamente de 1 a 6.

Imagem do menu da aplicação

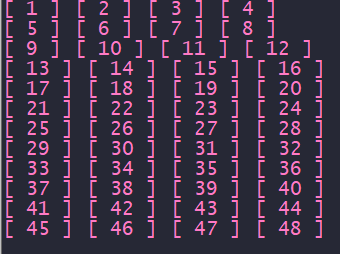


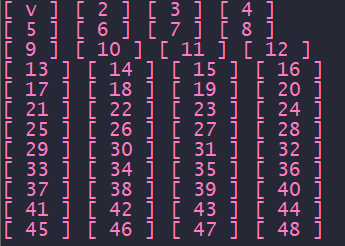


**4.2 Funcionalidades**

4.2.1 Poltronas Vendidas

Quando a opção selecionada for a, um (1), será exibido na tela uma lista com todas as poltronas numeradas de 1 a 48 ([ 1 ]), quando um lugar já estiver ocupado, o valor da poltrona será substituído pela letra V ([ V ]).



 Lista de poltronas

Lista de poltronas vendidas

Foi utilizado duas estruturas de repetição *for,* para realizar a construção dessa lista, o primeiro tem a função numerar todos os lugares, o segundo loop *for* tem o objetivo de exibir todas as poltronas disponíveis e ocupadas.

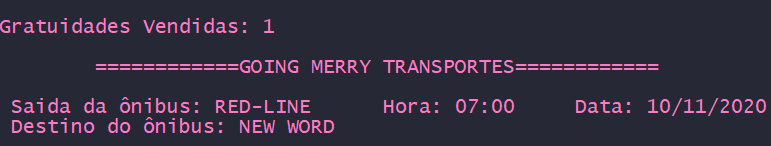




Foi utilizado dois vetores, *cadeiras[12][4]* e *cadeirasOcp[12][4]*, o vetor *cadeiras*, foi utilizado para controlar a quantidade total de poltronas, sem considerar quais foram vendidas, o vetor *cadeirasOcp*, foi usado para controlar a quantidade de poltronas vendidas.

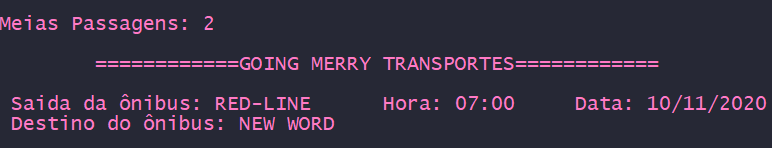
4.2.2 Gratuidade vendidas

Quando a opção selecionada for a segunda (2), o total de passagens gratuitas, destinados a idosos, será exibido na tela.



Menu de gratuidades vendidas

4.2.3 Meias Passagens

 Quando a opção selecionada for a três (3), será exibido o total de meias passagens destinadas a estudantes.

Menu de meias passagens

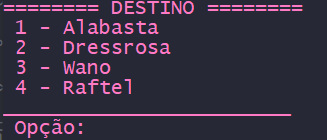
4.2.4 Vender Passagens

Ao selecionar a opção quatro (4), deverá selecionar o destino, tipo de passagem e a poltrona, onde todas as opções vão ser seguidas em sequência, e no final será exibido um ticket com todas as informações da venda.

4.2.4.1 Destino

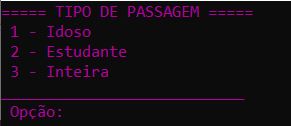
Ao selecionar a opção de vender Passagens, será exibida uma tela, com todos os destinos disponíveis, cada destino terá um código que deve ser inserido para selecionar a opção desejada.

No código fonte foi utilizado uma estrutura de Escolha-Caso (switch-Case), onde cada opção corresponde a um caso específico.



Menu de destino

Quando o destino for selecionado será necessário informar o tipo de passagem, sendo três tipos possíveis, gratuidade de idoso, apenas duas por ônibus, meia estudantil e inteira. Os valores de cada destino aumentam com o destino, onde a opção um sendo a mais barata e a quatro a mais cara.



Menu de tipos de passagem

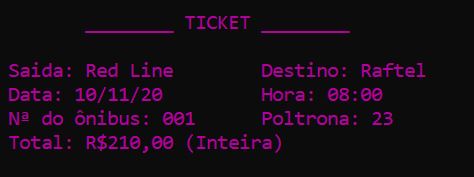


Acima está o código fonte do tipo de passagem idoso, segue alguns passos para validar a compra da passagem. Primeiro é verificado se tem passagens do tipo idoso disponíveis, e depois pede que seja informado o lugar que deseja comprar.



Menu de informe de Lugar

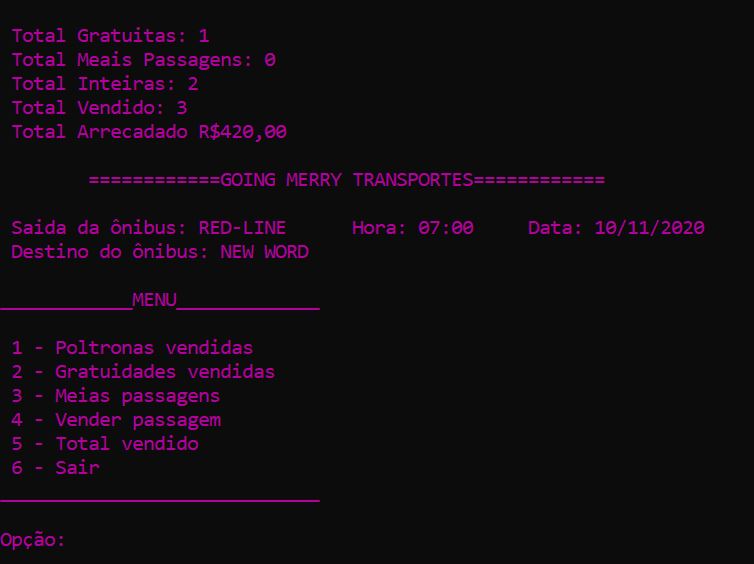
Em seguida é verificado se a cadeira informada é válida, estando entre 1 e 48, e verificado se essa poltrona está disponível, se estiver disponível e for válida, ela é vendida e imitido o ticket de compra.



Ticket de venda

4.2.5 Total Vendido

Ao selecionar a quinta opção será exibido o controle de vendas do dia, contendo as seguintes informações. Total de gratuidades vendidas, total de meias passagens vendidas, total de passagens inteiras, total de todos os tipos de passagens vendidas e o total arrecado em reais.



Menu da opção 5

4.2.6 Sair

A opção seis termina o programa. Por ser um programa executado na memória flash, logo não possui um arquivo de banco de dados, ele não guarda informações, tudo que foi feito vai ser perdido.

**CONCLUSÃO**

Este trabalho apresentou a importância da matéria de Engenharia de software e Técnicas de linguagem de programação, permitindo criar um sistema funcional para a empresa Going Marry Transportes, utilizando as técnicas e tecnologias apresentadas na sala de aula.

**REFERÊNCIAS**

ISO/IEC 9899:2018. ISO (em inglês). Consultado em 8 de novembro de 2018

History of the C Programming Language (em inglês). livinginternet.com. Consultado em 23 de maio de 2010

Dev-C++». Techtudo. Consultado em 24 de março de 2020

PINHO, Márcio Sarroglia. Utilização do DevC++.

Fairbairn, Andrew (19 de novembro de 2012). O Manual Dev-C++ e wxDev-C++: DevCpp.

Davis, Stephen R. (3 de maio de 2004). C++ For Dummies (em inglês). John Wiley & Sons. p. 12

C++ Grundbog (em dinamarquês). [S.l.]: Libris Media A/S. 2002. p. 13

Download Visual Studio Code. code.visualstudio.com (em inglês). Microsoft.

Microsoft Software License Terms. code.visualstudio.com (em inglês). Microsoft.

****

THIAGO DE OLIVEIRA MARTINS

**PROJETO DE REDE FÍSICO E LÓGICO**

**BELÉM**

**2020**

****

**PROJETO DE REDE FÍSICO E LÓGICO**

Trabalho elaborado pela aluna: THIAGO DE OLIVEIRA MARTINS, UP20214389, apresentado ao curso de Análise e Desenvolvimento de Sistemas, da Universidade Paulista (UNIP), referente ao 2° semestre do curso de nível tecnológico superior, sob a orientação do professor: Rodney Victor

**BELÉM**

**20****20**

SUMÁRIO

**RESUMO**....................................................................................................................**23**

**ABSTRACT**................................................................................................................**24**

**INTRODUÇÃO**...........................................................................................................**25**

**DESENVOLVIMENTO**...............................................................................................**26**

**CONCLUSÃO**............................................................................................................**32**

**REFERÊNCIAS**..........................................................................................................**33**

**RESUMO**

Este trabalho tem como objetivo, elaborar uma rede para a empresa Pesquisa Certa (empresa fictícia), onde será montado todo a parte logica e física para implantar a rede na nova sede da empresa. Uma rede bem configurada é muito importante para estabelecer uma comunicação eficiente em qualquer instituição, por esse motivo a este trabalho foca em apresentar os melhores equipamentos para o projeto, levando em conta o custo, desempenho e segurança. Além de montar toda a configuração de camada 3, da estrutura TCP/IP.

Palavras chaves: TCP/IP, rede, switch, host, comunicação

**ABSTRACT**

This work aims to develop a network for the company Pesquisa Certa (fictional company), where all the logical and physical part will be set up to implement the network in the new company headquarters. A well-configured network is very important to establish an efficient communication in any institution, for this reason this work focuses on presenting the best equipment for the project, taking into account the cost, performance and safety. In addition to assembling the entire layer 3 configuration, of the TCP / IP structure.

Keywords: TCP / IP, network, switch, host, communication

**INTRODUÇÃO**

O objetivo da implantação de uma estrutura de rede em uma empresa basicamente é garantir que todos os recursos de informações sejam compartilhados rapidamente, com segurança e de forma confiável. Dessa forma a rede deve possuir meios de transmissões eficientes, regras básicas (protocolos) e mecanismos capazes de garantir o transporte das informações entre os seus elementos constituintes.

Com o constante avanço tecnológico, o sistema de comunicação é indispensável em órgãos públicos ou privados, dessa forma é necessária uma estrutura de rede, onde sua estrutura física\ logica estarão sempre interligados para o tal funcionamento, pois dessa forma a agilidade no atendimento ao público torna –se muito melhor. Para que a comunicação tenha uma ótima oralidade é necessária uma estrutura de rede bem projetada, organizada, documentada e padronizada de conectores e meios de transmissão para redes de informática e telefonia. Permitindo a ligação de uma rede de servidores, estações de trabalho, impressoras, telefones, switches, hub e roteadores.

Com relação ao método, será utilizado o método dedutivo, pois o mesmo nos permite partir de leis mais gerais para esclarecer o problema proposto por este projeto.

A empresa Pesquisa Certa é uma empresa que atua no ramo de pesquisa de mercado, ampliará seu campo de atuação com a implantação e criação de uma nova base de operações no município de Jundiaí, dessa forma é necessário a implantação de um projeto físico\logico para a nova unidade respeitando suas exigências, onde além de ampliar a sua participação de mercado, irá gerar mais novos empregos no município. A nova base de operações da empresa contara inicialmente com cinquenta posições de atendimento ativo\passivo, um servidor de banco de dados, um servidor de aplicações.

**DESENVOLVIMENTO**

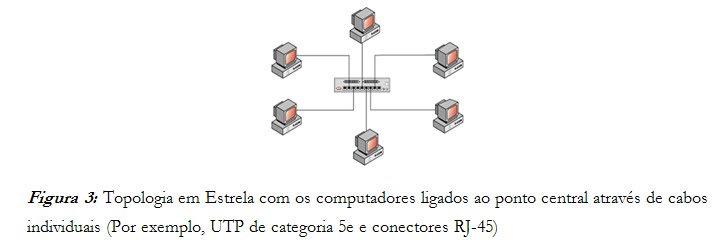
A empresa pesquisa certa, fez uma solicitação para criação de um projeto de rede para sua nova sede, essa rede deve ter como objetivo, a comunicação entre 55 hosts, incluindo dois servidores, um destinado para o banco de dados e outro como servidor de aplicação. A rede além da capacidade solicitada, deve ter suporte para um aumento de até 20%, em até 4 anos.

Outra parte muito importante foi a configuração logica da terceira camada de rede TCP/IP, fazendo a configuração dos endereços IP de todos os hosts, assim como o IP de rede, broadcast e gateway.

Para a instalação dessa rede, foi utilizada o tipo de rede LAN (Local Area Network), esse tipo de rede é de fácil instalação e de custo baixo, por ser uma rede local, além de poder fazer transferências de arquivos, compartilhar recursos como impressoras e bancos de dados.

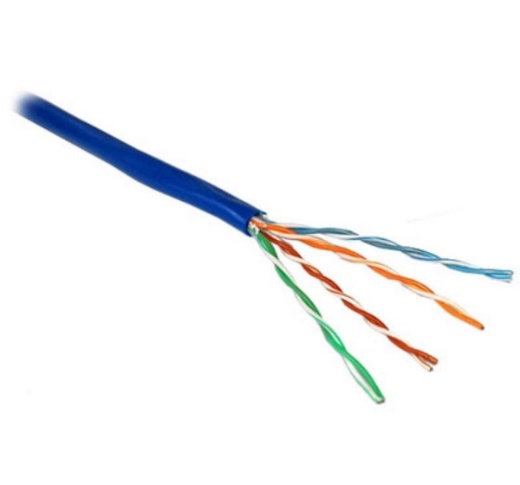
**Topologia**

A topologia usada na instalação foi a estrela, que consiste em ter um nó no centro, que é um dispositivo central (Hub ou switch) que pode ser ativo ou passivo, ao qual se ligam os vários computadores da rede, através de cabos individuais.



**Instalação física**

Para fazer a ligação dos equipamentos foi utilizado o cabo CAT5e junto com o conector RJ45, que é o mais utilizado para instalação de redes, por ter um excelente custo-benefício, por ter compatibilidade com a maiorias dos equipamentos.



Cabo CAT5e



Conector RJ45

Foi contratado um link dedicado de 400 mega, visando a uma comunicação instável, tanto no envio de dados quanto no recebimento. Visando ainda mais a instabilidade da comunicação, foi utilizado um segundo link de 100 mega, para casos de falha do link principal.

Para fazer a ligação da rede LAN foi utilizado um roteador, Cisco RV260 VPN Router, que oferecem tecnologia de rede privada (VPN) para que os funcionários remotos possam se conectar à sua rede através de um caminho seguro da internet.



Cisco RV260 VPN Router

As conexões dos hosts foram feitas pelo switch, Cisco SG250-50P 50-Port Gigabit POE Smart, pois oferece várias opções de gerenciamento, recursos sofisticados de segurança, qualidade de serviços (QoS) e recurso de roteamento estático da camada 3 muito além dos switches não gerenciáveis ou os de nível consumidor. Além de possuir uma interface web fácil de usar, aplicativos de rede inteligente e recurso Power over Ethernet Plus (PoE+).



Cisco RV260 VPN Router

Além dos switches foi utilizado um access point, UBIQUITI UNIFI AP-AC-LITE, por possuir conectividade 5.0 GHZ, e um sistema gerenciável de fácil instalação, com um alcance superior aos que estão no mercado.



UBIQUITI UNIFI AP-AC-LITE

**Configuração de camada 3**

A camada 3 da arquitetura TCP/IP, é uma das mais importantes, pois ela é a responsável pelo endereçamento IP, que funciona como um enderenço de uma casa, por exemplo, onde ela informa qual rede a máquina está localizada e qual é seu número.

Esse endereçamento é chamando Internet Protocol (IP), onde cada micro recebe apenas um endereço IP único que o identifica na rede. O endereço IP é composto por uma sequência de 32 bits, divididos em 4 grupos de 8 bits cada. Cada grupo de 8 bits recebe o nome de octeto.

O endereço IP é dividido em duas partes, uma identifica a rede em que o computador está conectado, e a segunda identifica o host dentro da rede. Esses endereços foram divididos em 5 classes, denominadas A, B, C, D, E, sendo que as três primeiras estão destinadas para endereçamento e as outras duas para expansões futuras.

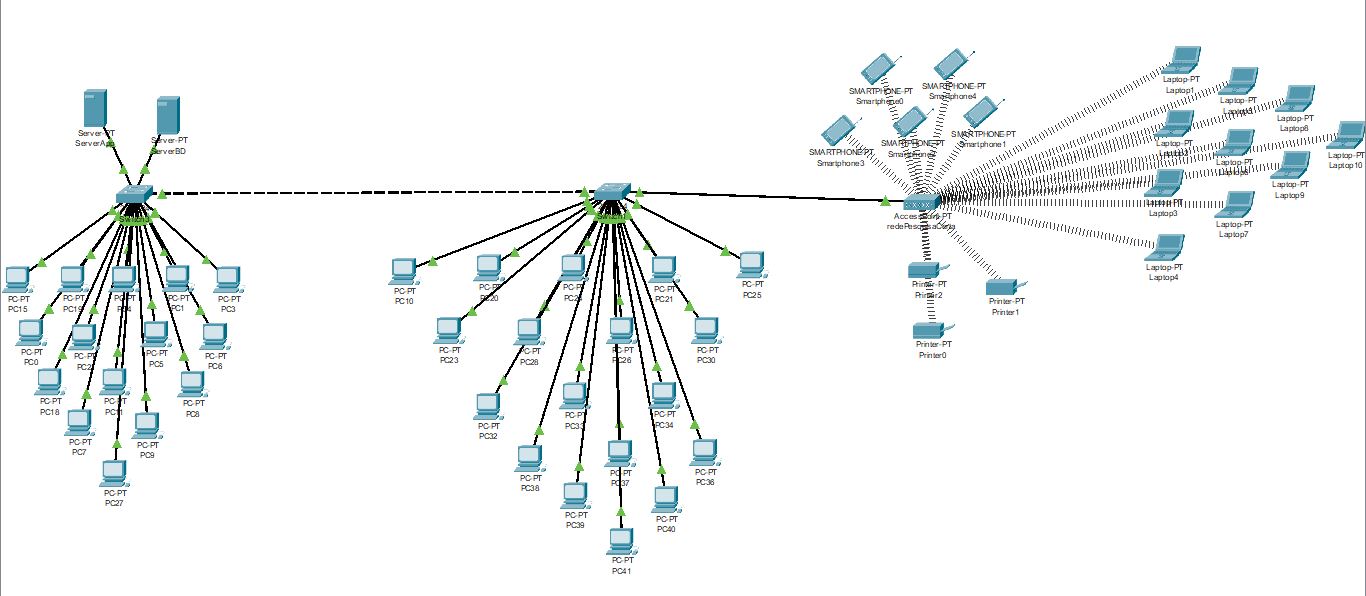
As classes são formas de identificar as divisões da rede, por isso cada classe tem uma mascará de rede, que define quantos bits vão ser usados para identificar a rede, e quantos bits são usados para identificar os hosts.

Para fazer o endereçamento dos hosts foi utilizado redes de classe C, que tem como máscara de rede padrão, 255.255.255.0, onde os três primeiros octetos são para a identificação da rede, e o ultimo octeto é para os hosts.

Por padrão existe alguns endereços reservados na rede, o primeiro é o endereço da rede, que por padrão é o primeiro número de IP disponível, que no caso da rede criado é o 192.168.0.0, esse endereço é reservado pois ele identifica a nossa rede. O segundo endereço reservado é o de broadcast, esse endereço é sempre o ultimo disponível, no caso o número 192.168.0.255. Outro que também deve ser reservado é o endereço de gateway, que no caso foi escolhido o número de IP 192.168.0.1.

Os servidores de aplicação e banco de dados, são reservados para eles os primeiros IPs disponíveis para hosts, sendo respectivamente o 192.168.0.2 e 192.168.0.3 para cada um deles.

A sequência de IP foi feita de forma que preenchesse todos os hosts solicitados da empresa, sendo o último o de número 192.168.0.54, podendo ainda expandir a rede até o endereço 192.168.0.255 por ser uma rede de classe C e possuir até 255 hosts.



**Ética e legislação**

A empresa pesquisa certa, classifica-se no âmbito das empresas de pequeno porte. Sendo assim podemos levantar alguns pontos consideráveis na área trabalhista, amparados pela constituição federal, consolidação das leis trabalhistas e convenção coletiva de trabalho, tais como: direito sindical, negociação e direito coletivo, responsabilidade social, código de ética essenciais para manter empresas em posicionamento no mercado.

Das obrigações trabalhistas das microempresas e empresas de pequeno porte, destacam-se as particularidades:

Em seu art.51 a lei complementar 123/2006 traz significativas simplificações das relações de trabalho, para as microempresas e as empresas de pequeno porte, dispensadas de diversos deveres acessórios, como:

- Afixação de quadro de horário de trabalho dos empregados, exceto do menor;

- Anotações das férias dos empregados em livros ou ficha de registro no momento da concessão, no entanto, por ocasião da rescisão do contrato de trabalho, a anotação deve ser feita;

- Manutenção do livro de inspeção do trabalho; empregar e matricular menores de 18 anos (aprendizes) nos cursos especializados mantidos pelo SENAI.

As demais obrigações trabalhistas permanecem sendo obrigadas, em especial:

- Anotações na carteira de trabalho e providencia social – CTPS;

- Arquivamento de documentos comprobatórios de cumprimento das obrigações trabalhistas e previdenciárias, enquanto não prescrevem essas obrigações;

- Apresentações das guias de recolhimento do fundo de garantia do tempo de serviço e informações à previdência social

- Apresentação das relações anuais de empregados e da relação anual de informações sociais e do cadastro geral de empregados e desempregados.

No âmbito do direito sindical, negociação e direito coletivo ressalte-se a contribuição sindical, que é uma obrigação tributária principal, devida por todas as pessoas jurídicas, autônomos, profissionais liberais e empregados aos respectivos sindicatos da categoria a que se enquadrem nos termos estabelecido pela consolidação das leis trabalhistas.

No que diz respeito a questões trabalhistas, ética empresarial se relaciona com questões morais relativas a seus funcionários, clientes e fornecedores. Essas questões são muito importantes, pois tem o poder de causar dano à imagem de uma empresa e destruir a credibilidade de seus funcionários (TURBAN; MCLEAN; WETHERBY,2004, P.34).

**CONCLUSÃO**

Este trabalho apresentou a importância de se ter uma rede para estabelecer comunicação de dados em uma empresa, e como cada a camada 3 da estrutura TCP/IP é de extrema importância, pois nela é feito o endereçamento IP, que serve de endereço para que os pacotes de dados sejam levados para os destinos certos e com segurança.

A ética e legislação também são de suma importância, pois elas definem o que se deve fazer, o que não se pode fazer além das diretrizes trabalhistas e legais, estabelecidas para a segurança dos clientes e de quem realiza a o projeto.

https://html1-f.scribdassets.com/5fo17lgxs037wckr/images/24-064e2adcce.jpg

# 

**REFERÊNCIAS**

https://www.cnstore.com.br/roteador-cisco-rv260-vpn-ro/

https://www.cnstore.com.br/switch-cisco-sg250-50p-50-port-gigabit-poe-smart/

https://www.ui.com/unifi/unifi-ap-ac-lite/

https://www.hardware.com.br/livros/linux-redes/capitulo-entendendo-enderecamento.html#:~:text=Cada%20grupo%20de%208%20bits,o%20host%20dentro%20da%20rede.

https://sites.google.com/site/disciplinadeiccr/Home/topologias-logicas-e-topologias-fisicas/topologia-em-estrela-ou-star