Eu preciso que tu me ajude a terminar o questionario de cadastro do motorista, ta faltando a parte de documentos. Vou te mandar tudo que eu acho que tu pode precisar (se precisar de mais algo pede) e tu adiciona o que precisar, sem editar nada. O que tu adicionar ou der uma leve modificada, sinalize com comentarios no meio do código (cada código que tu editar, me remande o código inteiro + a modificação, não só a adição para eu fazer)
Eu quero que esses documentos sejam adicionados no frontend, pelo usuario(motorista, nesse caso), como arquivo pdf, convertendo para a nomenclatura do arquivo e sendo salva em uploads (ex.: public/uploads/arquivo.pdf)
Vou te mandar o que eu acho que pode servir de usuarios também(seriam os pacientes), porque fiz isso que eu to pedindo para as imagens de perfil
Banco de dados:
(documentos ta como foreign key em motoristas, pode fazer o questionario da mesma forma que foi feita para endereços, que também é chave estrangeira em motoristas)

```
CREATE TABLE motoristas (
    cod INT NOT NULL AUTO_INCREMENT,
    matricula CHAR(4) UNIQUE,
    img VARCHAR(50),
    nome VARCHAR(40),
    data_nasc DATE,
    CPF CHAR(14) UNIQUE,
    fone CHAR(11) NOT NULL,
    email VARCHAR(30),
    endereco INT,
    genero INT,
    docs INT,
    senha VARCHAR(16) NOT NULL,
    CONSTRAINT pk_motoristas PRIMARY KEY (cod),
    CONSTRAINT fk_motoristas_endereco FOREIGN KEY (endereco) REFERENCES
enderecos(cod)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_motoristas_genero FOREIGN KEY (genero) REFERENCES
generos(cod)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_motoristas_docs FOREIGN KEY (docs) REFERENCES
documentos(cod)
        ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE documentos (
    cod INT NOT NULL AUTO_INCREMENT,
    carteira_trab VARCHAR(50),
    cursos VARCHAR(50),
```

```
    habilitacao VARCHAR(50),
    comprov_resid VARCHAR(50),
    comprov_escola VARCHAR(50),
    titulo_eleitor VARCHAR(50),
    ant_crim VARCHAR(50),
    exame_tox VARCHAR(50),
    CONSTRAINT pk_documentos PRIMARY KEY (cod)
);
CREATE TABLE usuarios (
    cod INT NOT NULL AUTO_INCREMENT,
    img VARCHAR(50),
    nome VARCHAR(40) NOT NULL,
    data_nasc DATE NOT NULL,
    CPF CHAR(14) UNIQUE NOT NULL,
    genero INT NOT NULL,
    email VARCHAR(30) NOT NULL,
    fone CHAR(11) NOT NULL,
    endereco INT NOT NULL,
    SUS CHAR(15) NOT NULL,
    senha VARCHAR(16) NOT NULL,
    CONSTRAINT pk_usuarios PRIMARY KEY (cod),
    CONSTRAINT fk_usuarios_endereco FOREIGN KEY (endereco) REFERENCES
enderecos(cod)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_usuarios_genero FOREIGN KEY (genero) REFERENCES
generos(cod)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Rotas:

adminRoutes: `const express = require('express');`

```
const router = express.Router();
const multer = require('multer');
const adminController = require('../controllers/adminController');

// Configuração do Multer
const storage = multer.diskStorage({
    destination: (req, file, cb) => cb(null, 'public/uploads/'),
    filename: (req, file, cb) => cb(null, Date.now() + ".jpg")
});
```

```
const upload = multer({ storage });

router.get('/usuarios/index', adminController.renderUsuarios);
router.get('/usuarios/buscar', adminController.buscarUsuarios);
router.delete('/usuarios/deletar/:cod', adminController.deletarUsuarios);
router.get('/usuarios/editar/:cod', adminController.editarUsuario);
router.put('/usuarios/editar/:cod', upload.single('foto_perfil'),
adminController.salvarEdicaoUsuario);

router.get('/motoristas/index', adminController.renderMotoristas);
router.get('/motoristas/buscar', adminController.buscarMotoristas);
router.get('/motoristas/editar/:cod', adminController.editarMotorista);
router.put('/motoristas/editar/:cod', upload.single('foto_perfil'),
adminController.salvarEdicaoMotorista);
router.delete('/motoristas/deletar/:cod',
adminController.deletarMotoristas);

router.get('/viagens/index', adminController.renderViagens);
router.get('/viagens/buscar-eventos',
adminController.renderBuscarEventos);
router.get('/viagens/nova-viagem', adminController.renderNovaViagem);
router.post('/viagens/add-nova-viagem',
adminController.renderCadastrarViagem);


router.get('/solicitacoes/index', adminController.renderSolicitacoes);

module.exports = router;
```

authRoutes.js: `const express = require('express');`

```
const router = express.Router();
const multer = require('multer');
const authController = require('../controllers/authController');

// Configuração do Multer
const storage = multer.diskStorage({
    destination: (req, file, cb) => cb(null, 'public/uploads/'),
    filename: (req, file, cb) => cb(null, Date.now() + ".jpg")
});
```

```
const upload = multer({ storage });

// Rotas
router.get('/entrada', authController.renderEntrada);
router.get('/cadastro', authController.renderCadastro);
router.get('/cadastro-sucesso', authController.renderCadastroSucesso);
router.post('/add-usuario', upload.single('foto_perfil'),
authController.cadastrarUsuario);

router.get('/cadastro-motorista', authController.renderCadastroMotorista);
router.post('/add-motorista', upload.single('foto_perfil'),
authController.cadastrarMotorista);

module.exports = router;
```

Models:

documento.js: `const db = require('./db');`

```
const Documento = db.sequelize.define('documento', {
  cod: {
    type: db.Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true,
    allowNull: false
  },
  carteira_trab: {
    type: db.Sequelize.STRING(50),
  },
  cursos: {
    type: db.Sequelize.STRING(50),
  },
  habilitacao: {
    type: db.Sequelize.STRING(50),
  },
  comprov_resid: {
    type: db.Sequelize.STRING(50),
  },
  comprov_escola: {
    type: db.Sequelize.STRING(50),
  },
```

```
  titulo_eleitor: {
    type: db.Sequelize.STRING(50),
  },
  ant_crim: {
    type: db.Sequelize.STRING(50),
  },
  exame_tox: {
    type: db.Sequelize.STRING(50),
  }
}, {
  timestamps: false
});

module.exports = Documento;
```

motorista.js: `const db = require('./db');`

```
const Endereco = require('./Endereco');
const Genero = require('./Genero');
const Documento = require('./Documento');

const Motorista = db.sequelize.define('Motorista', {
  cod: {
    type: db.Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true,
    allowNull: false
  },
  matricula: {
    type: db.Sequelize.STRING(4),
    unique: true,
    allowNull: false
  },
  img: {
    type: db.Sequelize.STRING(50),
  },
  nome: {
    type: db.Sequelize.STRING(40),
    allowNull: false
  },
  data_nasc: {
```

```
    type: db.Sequelize.DATEONLY,
    allowNull: false
  },
  CPF: {
    type: db.Sequelize.CHAR(14),
    allowNull: false,
    unique: true
  },
  fone: {
    type: db.Sequelize.CHAR(16),
    allowNull: false
  },
  email: {
    type: db.Sequelize.STRING(30),
    allowNull: false
  },
  enderecoID: {
    type: db.Sequelize.INTEGER,
    allowNull: false,
    field: 'endereco',
    references: {
      model: Endereco,
      key: 'cod'
    }
  },
  generoID: {
    type: db.Sequelize.INTEGER,
    allowNull: false,
    field: 'genero',
    references: {
      model: Genero,
      key: 'cod'
    }
  },
  docsID: {
    type: db.Sequelize.INTEGER,
    allowNull: false,
    field: 'docs',
    references: {
      model: Documento,
```

```
      key: 'cod'
    }
  },
  senha: {
          type: db.Sequelize.STRING(16),
          allowNull: false
  }
}, {
  timestamps: false
});

module.exports = Motorista;
```

usuario.js: `const db = require('./db');`

```
const Endereco = require('./Endereco');
const Genero = require('./Genero');

const Usuario = db.sequelize.define('Usuario', {
    cod: {
        type: db.Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        allowNull: false
    },
    img: {
        type: db.Sequelize.STRING(50),
    },
    nome: {
        type: db.Sequelize.STRING(40),
        allowNull: false
    },
    data_nasc: {
        type: db.Sequelize.DATEONLY,
        allowNull: false
    },
    CPF: {
        type: db.Sequelize.CHAR(14),
        allowNull: false,
        unique: true
    },
```

```
    generoID: {
        type: db.Sequelize.INTEGER,
        allowNull: false,
        field: 'genero', //nome de verda no banco
        references: {
            model: Genero,
            key: 'cod' }
    },
    email: {
        type: db.Sequelize.STRING(30),
        allowNull: false
    },
    fone: {
        type: db.Sequelize.CHAR(16),
        allowNull: false
    },
    enderecoID: {
        type: db.Sequelize.INTEGER,
        allowNull: false,
        field: 'endereco', //nome de verda no banco
        references: {
            model: Endereco,
            key: 'cod'
        }
    },
    SUS: {
        type: db.Sequelize.CHAR(15),
        allowNull: false
    },
    senha: {
        type: db.Sequelize.STRING(16),
        allowNull: false
    }
},{
  timestamps: false
  });

module.exports = Usuario;
```

index.js:

```
// models/index.js
const db = require('./db');
```

```
const Usuario = require('./Usuario');
const Endereco = require('./Endereco');
const Genero = require('./Genero');
const Status = require('./Status');
const Solicitacao = require('./Solicitacao');
const Motorista = require('./Motorista');
const Documento = require('./Documento');
const Viagem = require('./Viagem');
const Chefe = require('./Chefe');

//usuário
Usuario.belongsTo(Endereco, {
    foreignKey: 'enderecoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Endereco.hasMany(Usuario, {
    foreignKey: 'enderecoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

Usuario.belongsTo(Genero, {
    foreignKey: 'generoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Genero.hasMany(Usuario, {
    foreignKey: 'generoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

//solicitação
Solicitacao.belongsTo(Usuario, {
    foreignKey: 'usuarioID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Usuario.hasMany(Solicitacao, {
```

```
    foreignKey: 'usuarioID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

Solicitacao.belongsTo(Status, {
    foreignKey: 'statusID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Status.hasMany(Solicitacao, {
    foreignKey: 'statusID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

//motorista papai França <3
Motorista.belongsTo(Endereco, {
    foreignKey: 'enderecoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Endereco.hasMany(Motorista, {
    foreignKey: 'enderecoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

Motorista.belongsTo(Genero, {
    foreignKey: 'generoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Genero.hasMany(Motorista, {
    foreignKey: 'generoID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

Motorista.belongsTo(Documento, {
```

```
    foreignKey: 'docsID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Documento.hasMany(Motorista, {
    foreignKey: 'docsID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

//viagenss
Viagem.belongsTo(Motorista, {
    foreignKey: 'motoristaID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Motorista.hasMany(Viagem, {
    foreignKey: 'motoristaID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

Viagem.belongsTo(Status, {
    foreignKey: 'statusID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});
Status.hasMany(Viagem, {
    foreignKey: 'statusID',
    onDelete: 'CASCADE',
    onUpdate: 'CASCADE'
});

//sincroniza as tabelas no banco
db.sequelize.sync({force: false})
    .then(() => {
        console.log('Tabelas sincronizadas com sucesso.');
    })
    .catch((err) => {
        console.error('Erro ao sincronizar as tabelas:', err);
```

```
    });

console.log('Arquivo models/index.js executado');
module.exports = { db, Usuario, Endereco, Genero, Status, Solicitacao,
Motorista, Documento, Viagem, Chefe};
```

Controllers:

adminController.js: `const { Usuario, Endereco, Genero, Motorista, Documento, Viagem } = require('../models');`

```
const { Op } = require('sequelize');

exports.renderUsuarios = async (req, res) => {
  try {
    const posts = await Usuario.findAll({
      include: [
        { model: Endereco },
        { model: Genero }
      ]
    });
    posts.sort((a, b) => b.cod - a.cod);
    res.render('admin/usuarios/index', {
      posts,
      layout: 'layouts/layoutAdmin',
      paginaAtual: 'usuarios'
    });
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao buscar usuários: ' + erro);
  }
};

exports.buscarUsuarios = async (req, res) => {
  try {
    const termo = req.query.q || '';
    const usuarios = await Usuario.findAll({
      where: {
        nome: {
          [Op.like]: `%${termo}%`
        }
      },
```

```
    });
    res.json(usuarios);
  } catch (erro) {
    console.error(erro);
    res.status(500).json({ erro: 'Erro ao buscar usuários' });
  }
};

exports.deletarUsuarios = async (req, res) => {
  try {
    const { cod } = req.params;
    const usuario = await Usuario.findByPk(cod);
    if (!usuario) {
      return res.status(404).send('Usuário não encontrado');
    }
    await usuario.destroy();
    res.redirect('/admin/usuarios/index');
  } catch (error) {
    console.error('Erro ao deletar usuário:', error);
    res.status(500).send('Erro interno no servidor');
  }
};

exports.editarUsuario = async (req, res) => {
  try {
    const cod = req.params.cod;
    const usuario = await Usuario.findOne({
      where: { cod },
      include: [
        { model: Endereco },
        { model: Genero }
      ]
    });
    if (!usuario) {
      return res.status(404).send('Usuário não encontrado');
    }
    res.render('admin/usuarios/editar', {
      usuario,
      layout: 'layouts/layoutAdmin',
      paginaAtual: 'usuarios'
```

```
    });
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao carregar usuário para edição');
  }
};

exports.salvarEdicaoUsuario = async (req, res) => {
  try {
    const cod = req.params.cod;
    const usuario = await Usuario.findByPk(cod);
    if (!usuario) {
      return res.status(404).send('Usuário não encontrado');
    }
    await Endereco.update({
      rua: req.body.rua,
      numero: req.body.numero,
      bairro: req.body.bairro,
      cidade: req.body.cidade,
      UF: req.body.uf,
      CEP: req.body.cep
    }, {
      where: { cod: usuario.enderecoID }
    });
    await Usuario.update({
      img: req.file ? req.file.filename : usuario.img,
      nome: req.body.nome,
      data_nasc: req.body.data_nasc,
      CPF: req.body.CPF,
      generoID: req.body.genero,
      email: req.body.email,
      fone: req.body.fone,
      SUS: req.body.SUS
    }, {
      where: { cod }
    });
    res.redirect('/admin/usuarios/index');
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao salvar edição: ' + erro);
```

```
  }
};

exports.renderMotoristas = async (req, res) => {
  try {
    const motoristas = await Motorista.findAll({
      include: [
        { model: Endereco },
        { model: Genero }
      ]
    });
    motoristas.sort((a, b) => b.cod - a.cod);
    res.render('admin/motoristas/index', {
      posts: motoristas,
      layout: 'layouts/layoutAdmin',
      paginaAtual: 'motoristas'
    });
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao buscar motoristas: ' + erro);
  }
};

exports.buscarMotoristas = async (req, res) => {
  try {
    const termo = req.query.q || '';
    const motoristas = await Motorista.findAll({
      where: {
        nome: {
          [Op.like]: `%${termo}%`
        }
      },
      include: [
        { model: Endereco },
        { model: Genero }
      ]
    });
    res.json(motoristas);
  } catch (erro) {
    console.error(erro);
```

```
    res.status(500).json({ erro: 'Erro ao buscar motoristas' });
  }
};

exports.deletarMotoristas = async (req, res) => {
  try {
    const { cod } = req.params;
    const motorista = await Motorista.findByPk(cod);
    if (!motorista) {
      return res.status(404).send('Motorista não encontrado');
    }
    await motorista.destroy();
    res.redirect('/admin/motoristas/index');
  } catch (error) {
    console.error('Erro ao deletar motorista:', error);
    res.status(500).send('Erro interno no servidor');
  }
};

exports.editarMotorista = async (req, res) => {
  try {
    const cod = req.params.cod;
    const motorista = await Motorista.findOne({
      where: { cod },
      include: [
        { model: Endereco },
        { model: Genero }
      ]
    });
    if (!motorista) {
      return res.status(404).send('Motorista não encontrado');
    }
    res.render('admin/motoristas/editar', {
      usuario: motorista,
      layout: 'layouts/layoutAdmin',
      paginaAtual: 'motoristas'
    });
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao carregar motorista para edição');
```

```
  }
};

exports.salvarEdicaoMotorista = async (req, res) => {
  try {
    const cod = req.params.cod;
    const motorista = await Motorista.findByPk(cod);
    if (!motorista) {
      return res.status(404).send('Motorista não encontrado');
    }

    const cpfVerificando = req.body.CPF.replace(/\D/g, '');
    const foneVerificando = req.body.fone.replace(/\D/g, '');

    await Endereco.update({
      rua: req.body.rua,
      numero: req.body.numero,
      bairro: req.body.bairro,
      cidade: req.body.cidade,
      UF: req.body.uf,
      CEP: req.body.cep
    }, {
      where: { cod: motorista.enderecoID }
    });

    await Motorista.update({
      img: req.file ? req.file.filename : motorista.img,
      nome: req.body.nome,
      data_nasc: req.body.data_nasc,
      CPF: cpfVerificando,
      generoID: req.body.genero,
      email: req.body.email,
      fone: foneVerificando,
      // docsID e senha não alterados
    }, {
      where: { cod }
    });

    res.redirect('/admin/motoristas/index');
  } catch (erro) {
```

```
    console.error(erro);
    res.status(500).send('Erro ao salvar edição: ' + erro);
  }
};

exports.renderSolicitacoes = async (req, res) => {
  try {
    res.render('admin/solicitacoes/index', {
      layout: 'layouts/layoutAdmin',
      paginaAtual: 'solicitacoes'
    });
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao carregar solicitações');
  }
};

exports.renderViagens = (req, res) => {
    res.render('admin/viagens/index', {
      layout: 'layouts/layoutAdmin',
      paginaAtual: 'viagens',
    });
};

exports.renderBuscarEventos = async (req, res) => {
    const viagens = await Viagem.findAll();

    const eventos = viagens.map(v => ({
      title: v.destino_cid,
      start: v.data_viagem
    }));
    res.json(eventos);
}

exports.renderNovaViagem = async (req, res) => {
  try {
    const motoristas = await Motorista.findAll();
    const dataSelecionada = req.query.data_viagem || '';
    res.render('admin/viagens/nova-viagem', {
      layout: 'layouts/layoutAdmin',
```

```
      paginaAtual: 'viagens',
      dataSelecionada,
      motoristas
    });
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao carregar motoristas: ' + erro);
  }
};

exports.renderCadastrarViagem = async (req, res) => {
  try {
    await Viagem.create({
      destino_cid: req.body.destino_cid,
      data_viagem: req.body.data_viagem,
      horario_saida: req.body.horario_saida,
      lugares_dispo: req.body.lugares_dispo,
      modelo_car: req.body.modelo_car,
      placa: req.body.placa,
      motoristaID: req.body.motoristaID,
      statusID: 1
    });
    res.redirect('/admin/viagens/index');
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao cadastrar viagem: ' + erro);
  }
};
```

authRoutes.js: `const Usuario = require('../models/Usuario');`

```
const Endereco = require('../models/Endereco');
const Motorista = require('../models/Motorista');
const Documento = require('../models/Documento');

exports.renderEntrada = (req, res) => {
  res.render('auth/entrada', { layout: 'layouts/layoutAuth' });
};

exports.renderCadastro = (req, res) => {
  res.render('auth/cadastro', {
```

```
    layout: 'layouts/layoutAuth',
    erroCPF: null,
    erroSUS: null,
    preenchido: {}
  });
};

exports.renderCadastroSucesso = (req, res) => {
  res.render('auth/cadastro-sucesso', { layout: 'layouts/layoutAuth' });
};

exports.cadastrarUsuario = async (req, res) => {
  try {
    const cpfVerificando = req.body.CPF.replace(/\D/g, '');
    const susVerificando = req.body.SUS.replace(/\D/g, '');
    const foneVerificando = req.body.fone.replace(/\D/g, '');

    const cpfExistente = await Usuario.findOne({ where: { CPF:
cpfVerificando } });
    const susExistente = await Usuario.findOne({ where: { SUS:
susVerificando } });

    const erros = {
      erroCPF: null,
      erroSUS: null,
      erroFone: null,
    };

    if (cpfExistente) erros.erroCPF = 'CPF já cadastrado.';
    else if (cpfVerificando.length !== 11) erros.erroCPF = 'CPF
inválido.';

    if (susExistente) erros.erroSUS = 'Número do SUS já cadastrado.';
    else if (susVerificando.length !== 15) erros.erroSUS = 'Número do SUS
inválido.';

    if (foneVerificando.length < 11) erros.erroFone = 'Telefone inválido.
Deve conter 11 dígitos.';

    // Verifica se tem algum erro com valor (não null, não vazio)
```

```
    const temErro = Object.values(erros).some(erro => erro && erro.length
> 0);

    if (temErro) {
      return res.render('auth/cadastro', {
        layout: 'layouts/layoutAuth',
        ...erros,
        preenchido: req.body
      });
    }

    // Criação do endereço
    const enderecoCriado = await Endereco.create({
      rua: req.body.rua,
      numero: req.body.numero,
      bairro: req.body.bairro,
      cidade: req.body.cidade,
      UF: req.body.uf,
      CEP: req.body.cep
    });

    // Criação do usuário
    await Usuario.create({
      img: req.file ? req.file.filename : null,
      nome: req.body.nome,
      data_nasc: req.body.data_nasc,
      CPF: cpfVerificando,
      generoID: req.body.genero,
      email: req.body.email,
      fone: foneVerificando,
      enderecoID: enderecoCriado.cod,
      SUS: susVerificando,
      senha: req.body.senha
    });

    // Sucesso
    res.redirect('/auth/cadastro-sucesso');

  } catch (erro) {
    console.error(erro);
```

```
    res.status(500).send('Erro ao cadastrar: ' + erro);
  }
};

exports.renderCadastroMotorista = (req, res) => {
  res.render('auth/cadastro-motorista', {
    layout: 'layouts/layoutAuth',
    erroCPF: null,
    erroMatricula: null,
    preenchido: {},
    erroFone: null
  });
};

exports.cadastrarMotorista = async (req, res) => {
  try {
    const cpfVerificando = req.body.CPF.replace(/\D/g, '');
    const matriculaVerificando = req.body.matricula.replace(/\D/g, '');
    const foneVerificando = req.body.fone.replace(/\D/g, '');

    const cpfExistente = await Motorista.findOne({ where: { CPF:
cpfVerificando } });
    const matriculaExistente = await Motorista.findOne({ where: {
matricula: matriculaVerificando } });

    const erros = {
      erroCPF: null,
      erroMatricula: null,
      erroFone: null,
    };

    if (cpfExistente) erros.erroCPF = 'CPF já cadastrado.';
    else if (cpfVerificando.length !== 11) erros.erroCPF = 'CPF
inválido.';

    if (matriculaExistente) erros.erroMatricula = 'Matrícula já
cadastrada.';
    else if (!/^\d{4}$/.test(matriculaVerificando)) erros.erroMatricula =
'Matrícula inválida. Deve conter exatamente 4 dígitos.';
```

```
    if (foneVerificando.length < 11) erros.erroFone = 'Telefone inválido.
Deve conter 11 dígitos.';

    const temErro = Object.values(erros).some(erro => erro);

    if (temErro) {
      return res.render('auth/cadastro-motorista', {
        layout: 'layouts/layoutAuth',
        ...erros,
        preenchido: req.body
      });
    }

    const enderecoCriado = await Endereco.create({
      rua: req.body.rua,
      numero: req.body.numero,
      bairro: req.body.bairro,
      cidade: req.body.cidade,
      UF: req.body.uf,
      CEP: req.body.cep
    });

    const docsCriado = await Documento.create({});

    await Motorista.create({
      img: req.file ? req.file.filename : null,
      nome: req.body.nome,
      data_nasc: req.body.data_nasc,
      CPF: cpfVerificando,
      fone: foneVerificando,
      email: req.body.email,
      generoID: req.body.genero,
      enderecoID: enderecoCriado.cod,
      docsID: docsCriado.cod,
      senha: req.body.senha,
      matricula: matriculaVerificando
    });

    res.redirect('/admin/motoristas/index');
```

```
  } catch (erro) {
    console.error(erro);
    res.status(500).send('Erro ao cadastrar motorista: ' + erro);
  }
};
```

views/admin/motoristas:

Editar.ejs: `<div class="col-12 d-flex justify-content-center espaco-card-edit">`

```
  <div class="card p-3 shadow-sm mt-2" style="width: 100%; max-width: 720px; background-color: #d1e3fd;">
    <form action="/admin/motoristas/editar/<%= usuario.cod %>?_method=PUT" method="POST" enctype="multipart/form-data" class="form-bold">

      <div class="d-flex flex-row align-items-center mb-3">
        <label for="foto_perfil" class="me-3 ">
          <img id="preview_foto" src="<%= usuario.img ? '/uploads/' + usuario.img : '/assets/default-profile.jpg' %>"  alt="Foto de perfil" class="foto-perfil" />
        </label>
        <input type="file" id="foto_perfil" name="foto_perfil" accept="image/*" class="d-none" onchange="previewImagem(event)">
        <div class="flex-grow-1">
          <div class="d-flex align-items-center">
            <input type="text" id="nome" name="nome" maxlength="40" required class="form-control input-transparente" value="<%= usuario.nome %>" />
          </div>
        </div>
      </div>

      <div class="mb-3 d-flex align-items-center">
        <label for="data_nasc" class="me-2" >Data de nascimento:</label>
        <input type="date" id="data_nasc" name="data_nasc" required class="form-control input-transparente" style="width: 110px;" value="<%= usuario.data_nasc %>" />
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
```

```
        <label for="CPF" class="me-2" >CPF:</label>
        <input type="text" id="CPF" name="CPF" maxlength="14" required
class="form-control input-transparente" value="<%=
usuario.CPF.replace(/^(\d{3})(\d{3})(\d{3})(\d{2})$/, "$1.$2.$3-$4")  %>"
/>
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="genero" class="me-2" >Gênero:</label>
        <select id="genero" name="genero" required class="form-select
input-transparente" style="width: auto;">
          <option disabled>Não selecionado</option>
          <option value="1" <%= usuario.generoID == 1 ? 'selected' : ''
%>>Masculino</option>
          <option value="2" <%= usuario.generoID == 2 ? 'selected' : ''
%>>Feminino</option>
          <option value="3" <%= usuario.generoID == 3 ? 'selected' : ''
%>>Não-binário</option>
          <option value="4" <%= usuario.generoID == 4 ? 'selected' : ''
%>>Outro</option>
        </select>
      </div>

      <hr>

      <!-- Campo: Email -->
      <div class="mb-3 d-flex align-items-center">
        <label for="email" class="me-2" >Email:</label>
        <input type="email" id="email" name="email" maxlength="30"
required class="form-control input-transparente"  value="<%= usuario.email
%>" />
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="fone" class="me-2" >Telefone:</label>
```

```
        <input type="text" id="fone" name="fone" maxlength="16"
class="form-control input-transparente"  value="<%=
usuario.fone.replace(/^(\d{2})(\d{5})(\d{4})$/, "($1) $2-$3") %>" />
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="rua" class="me-2" >Rua:</label>
        <input type="text" id="rua" name="rua" required
class="form-control input-transparente" style="max-width: 300px;"
value="<%= usuario.endereco?.rua || '' %>" />
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="numero" class="me-2" >Número:</label>
        <input type="text" id="numero" name="numero" required
class="form-control input-transparente"  value="<%=
usuario.endereco?.numero || '' %>" />
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="bairro" class="me-2" >Bairro:</label>
        <input type="text" id="bairro" name="bairro" required
class="form-control input-transparente"  value="<%=
usuario.endereco?.bairro || '' %>" />
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="cidade" class="me-2" >Cidade:</label>
        <input type="text" id="cidade" name="cidade" required
class="form-control input-transparente" value="<%=
usuario.endereco?.cidade || '' %>" />
      </div>
```

```
      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="uf" class="me-2" >Estado (UF):</label>
        <select id="uf" name="uf" required class="form-select
input-transparente" style="width: 60px;">
          <option disabled>Não selecionado</option>
          <% ['SC', 'PR', 'RS', 'SP', 'RJ', 'MG', 'ES', 'DF', 'GO', 'MT',
'MS', 'AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE', 'AC', 'AP',
'AM', 'PA', 'RO', 'RR', 'TO'].forEach(estado => { %>
            <option value="<%= estado %>" <%= usuario.endereco?.UF ===
estado ? 'selected' : '' %>><%= estado %></option>
          <% }) %>
        </select>
      </div>

      <hr>

      <div class="mb-3 d-flex align-items-center">
        <label for="cep" class="me-2">CEP:</label>
        <input type="text" id="cep" name="cep" required
class="form-control input-transparente" value="<%=
usuario.endereco.CEP.replace(/^(\d{5})(\d{3})$/, '$1-$2') %>" />
      </div>

      <hr>

      <div class="d-flex justify-content-between">
        <a href="/admin/motoristas/index" class="botao">Cancelar</a>
        <button type="submit" class="botao">Salvar</button>
      </div>
    </form>
  </div>
</div>


<script>
  function previewImagem(event) {
  const input = event.target;
```

```
  const reader = new FileReader();

  reader.onload = function () {
    const preview = document.getElementById('preview_foto');
    preview.src = reader.result;
  };

  reader.readAsDataURL(input.files[0]);
}

 document.querySelector('form').addEventListener('submit', function (e) {
    const cpfInput = document.getElementById('CPF');
    const foneInput = document.getElementById('fone');
    const cepInput = document.getElementById('cep');

    cpfInput.value = cpfInput.value.replace(/\D/g, '');
    foneInput.value = foneInput.value.replace(/\D/g, '');
    cepInput.value = cepInput.value.replace(/\D/g, '');
  });

</script>
```

Index.ejs: `<div class="container mt-4">`

```
  <div class="row justify-content-center mb-5">
    <div class="d-flex justify-content-center align-items-center"
style="gap: 10px;">
      <input type="search"
      id="pesquisa"
      placeholder="Pesquisar motorista por nome ou CPF"
      class="form-control pesquisa-custom mb-3">
      <a href="/auth/cadastro-motorista" class="botao-cadastro">Cadastrar
Motorista</a>
    </div>

    <% posts.forEach(function(post) { %>

      <div class="col-12 d-flex justify-content-center">
        <div class="card p-3 shadow-sm mt-2" style="width: 100%;
max-width: 720px; background-color: #d1e3fd;">
          <div class="d-flex flex-row align-items-center">
```

```
            <img
              src="<%= post.img ? '/uploads/' + post.img :
'/assets/default-profile.jpg' %>"
              alt="Foto de perfil"
              class="foto-perfil me-3">

            <div class="flex-grow-1">
              <h5 class="nome-perfil mb-0"><%= post.nome %></h5>
            </div>

            <button class="botao ver-mais-btn">Ver mais</button>
            </div>

            <!--VER MAIS-->
            <div class="detalhes">
            <p><strong>Data de Nascimento: </strong><%=
post.data_nasc.split('-').reverse().join('/') %></p>
            <hr>
            <p><strong>CPF: </strong><span class="cpf-perfil"><%=
post.CPF.replace(/^(\d{3})(\d{3})(\d{3})(\d{2})$/, "$1.$2.$3-$4")
%></span></p>
            <hr>
            <p><strong>Gênero: </strong> <%= post.genero.descricao %></p>
            <hr>
            <p><strong>Email: </strong> <%= post.email %></p>
            <hr>
            <p><strong>Telefone: </strong><%=
post.fone.replace(/^(\d{2})(\d{5})(\d{4})$/, "($1) $2-$3")
            %></p>
            <hr>
            <p><strong>Endereço: </strong>
              <%= post.endereco?.rua || '' %>,
              <%= post.endereco?.numero || '' %> -
              <%= post.endereco?.bairro || '' %> -
              <%= post.endereco?.cidade || '' %>
              (<%= post.endereco?.UF || '' %>)
              CEP: <%= post.endereco?.CEP
                          ? post.endereco.CEP.replace(/^(\d{5})(\d{3})$/,
"$1-$2")
```

```
                        : ''
                    %>
            </p>
            <hr>

<div class="d-flex justify-content-end gap-3">
  <a href="/admin/motoristas/editar/<%= post.cod %>"
class="botao">Editar</a>

  <form action="/admin/motoristas/deletar/<%= post.cod %>?_method=DELETE"
method="POST">
    <button type="submit" class="botao bg-vermelho" onclick="return
confirm('Tem certeza que deseja deletar este motorista?')">
      Deletar
    </button>
  </form>
</div>

          </div>
        </div>
      </div>

    <% }) %>

  </div>
</div>

<script>
  const inputPesquisa = document.getElementById('pesquisa');
  const cards = Array.from(document.querySelectorAll('.card'));
  const row = document.querySelector('.row');

  cards.forEach((card, index) => {
    card.setAttribute('data-original-index', index);
  });

  inputPesquisa.addEventListener('input', function () {
    const termo = this.value.trim().toLowerCase();

    if (termo.length === 0) {
```

```
      cards.forEach(card => {
        card.style.display = 'block';
        card.setAttribute('data-relevancia', '0');
      });

      const fragment = document.createDocumentFragment();

      cards
        .sort((a, b) => a.getAttribute('data-original-index') -
b.getAttribute('data-original-index'))
        .forEach(card => fragment.appendChild(card));

      row.appendChild(fragment);

    } else {
      cards.forEach(card => {
        const nomeELemento = card.querySelector('.nome-perfil');
        const nome = nomeELemento ? nomeELemento.textContent.toLowerCase()
: '';

        const cpfElemento = card.querySelector('.cpf-perfil');
        const cpf = cpfElemento ?
cpfElemento.textContent.replace(/[.\-]/g, '') : '';

        if (nome.startsWith(termo) || cpf.startsWith(termo)) {
          card.style.display = 'block';
          card.setAttribute('data-relevancia', '1');
        } else if (nome.includes(termo) || cpf.includes(termo)) {
          card.style.display = 'block';
          card.setAttribute('data-relevancia', '2');
        } else {
          card.style.display = 'none';
          card.setAttribute('data-relevancia', '99');
        }

      });

      const fragment = document.createDocumentFragment();

      cards
```

```
        .filter(card => card.style.display === 'block')
        .sort((a, b) => a.getAttribute('data-relevancia') -
b.getAttribute('data-relevancia'))
        .forEach(card => fragment.appendChild(card));

      row.appendChild(fragment);
    }
  });

  // VER MAIS / VER MENOS
  document.querySelectorAll('.ver-mais-btn').forEach(btn => {
    btn.addEventListener('click', function () {
      const card = this.closest('.card');
      const detalhes = card.querySelector('.detalhes');
      const estaExpandido = detalhes.classList.contains('expandido');

      if (estaExpandido) {
        detalhes.style.height = '0px';
        detalhes.style.marginTop = '0';
        detalhes.classList.remove('expandido');
        this.textContent = 'Ver mais';
      } else {
        detalhes.style.height = detalhes.scrollHeight + 'px';
        detalhes.style.marginTop = '16px';
        detalhes.classList.add('expandido');
        this.textContent = 'Ver menos';
      }
    });
  });
</script>
```

views/auth/

Cadastro-motorista.ejs:

```
  <div class="card shadow rounded-4 p-5 mb-5" style="width: 100%;
max-width: 600px; background-color: #fff;">
    <form action="/auth/add-motorista" method="POST"
enctype="multipart/form-data">
      <div class="text-center mb-4">
```

```
        <img src="/assets/logo.png" alt="Logo" style="width: 50px; height:
70px;">
        <h5 class="mt-2 logo-nome-cadastro">Cadastrar motorista</h5>

        <label for="foto_perfil">
          <img id="preview_foto" src="/assets/default-profile.jpg"
alt="Foto de perfil"
            class="rounded-circle mt-3 mb-2"
            style="width: 90px; height: 90px; object-fit: cover; cursor:
pointer; border: 2px solid #ccc;" />
        </label>

        <input type="file" id="foto_perfil" name="foto_perfil"
accept="image/*" class="d-none" onchange="previewImagem(event)">
      </div>

      <div class="mb-3">
        <label for="nome" class="form-label">Nome</label>
        <input type="text" id="nome" name="nome" maxlength="40" required
class="form-control"
          placeholder="Digite o nome completo" value="<%= preenchido?.nome
|| '' %>" />
      </div>

      <div class="mb-3">
        <label for="data_nasc" class="form-label">Data de
nascimento</label>
        <input type="date" id="data_nasc" name="data_nasc" required
class="form-control"
          value="<%= preenchido?.data_nasc || '' %>" />
      </div>

     <div class="mb-3">
        <label for="CPF" class="form-label">CPF</label>
        <input type="text" id="CPF" name="CPF" maxlength="14" required
            class="form-control <% if (erroCPF) { %>is-invalid<% } %>"
placeholder="Digite o CPF"
            value="<%= preenchido?.CPF || '' %>" />
        <% if (erroCPF) { %>
            <div class="invalid-feedback"><%= erroCPF %></div>
```

```
        <% } %>
    </div>

      <div class="mb-3">
        <label for="genero" class="form-label">Gênero</label>
        <select id="genero" name="genero" required class="form-select">
          <option value="" disabled <%= !preenchido?.genero ? 'selected' :
'' %>>Não selecionado</option>
          <option value="1" <%= preenchido?.genero == '1' ? 'selected' :
'' %>>Masculino</option>
          <option value="2" <%= preenchido?.genero == '2' ? 'selected' :
'' %>>Feminino</option>
          <option value="3" <%= preenchido?.genero == '3' ? 'selected' :
'' %>>Não-binário</option>
          <option value="4" <%= preenchido?.genero == '4' ? 'selected' :
'' %>>Outro</option>
        </select>
      </div>

      <div class="mb-3">
        <label for="email" class="form-label">Email</label>
        <input type="email" id="email" name="email" maxlength="30"
required class="form-control"
          placeholder="Digite o e-mail" value="<%= preenchido?.email || ''
%>" />
      </div>

     <div class="mb-3">
        <label for="fone" class="form-label">Telefone</label>
        <input type="text" id="fone" name="fone" maxlength="15"
            class="form-control <% if (erroFone) { %>is-invalid<% } %>"
            placeholder="Digite seu telefone para contato"
            value="<%= preenchido?.fone || '' %>" />
        <% if (erroFone) { %>
            <div class="invalid-feedback"><%= erroFone %></div>
        <% } %>
    </div>

      <hr>
      <h6 style="color:#909090">Endereço</h6>
```

```
      <div class="mb-3">
        <label for="rua" class="form-label">Rua</label>
        <input type="text" id="rua" name="rua" required
class="form-control"
          placeholder="Rua da residência" value="<%= preenchido?.rua || ''
%>" />
      </div>

      <div class="mb-3">
        <label for="numero" class="form-label">Número</label>
        <input type="text" id="numero" name="numero" required
class="form-control"
          placeholder="Número da residência" value="<%= preenchido?.numero
|| '' %>" />
      </div>

      <div class="mb-3">
        <label for="bairro" class="form-label">Bairro</label>
        <input type="text" id="bairro" name="bairro" required
class="form-control"
          placeholder="Bairro da residência" value="<%= preenchido?.bairro
|| '' %>" />
      </div>

      <div class="mb-3">
        <label for="cidade" class="form-label">Cidade</label>
        <input type="text" id="cidade" name="cidade" required
class="form-control"
          placeholder="Cidade da residência" value="<%= preenchido?.cidade
|| '' %>" />
      </div>

      <div class="mb-3">
        <label for="uf" class="form-label">Estado (UF)</label>
        <select id="uf" name="uf" required class="form-select">
          <option disabled <%= !preenchido?.uf ? 'selected' : '' %>>Não
selecionado</option>
```

```
          <% const estados =
['SC','PR','RS','SP','RJ','MG','ES','DF','GO','MT','MS','AL','BA','CE','MA
','PB','PE','PI','RN','SE','AC','AP','AM','PA','RO','RR','TO']; %>
          <% estados.forEach(estado => { %>
            <option value="<%= estado %>" <%= preenchido?.uf === estado ?
'selected' : '' %>><%= estado %></option>
          <% }); %>
        </select>
      </div>

      <div class="mb-3">
        <label for="cep" class="form-label">CEP</label>
        <input type="text" id="cep" name="cep" required
class="form-control"
            placeholder="Digite o CEP" value="<%= preenchido?.cep || ''
%>" />
    </div>

      <hr>

      <div class="mb-3">
        <label for="matricula" class="form-label">Matrícula</label>
        <input type="text" id="matricula" name="matricula" maxlength="4"
required
          class="form-control <% if (erroMatricula) { %>is-invalid<% } %>"
placeholder="Matrícula com 4 dígitos"
          value="<%= preenchido?.matricula || '' %>" />
        <% if (erroMatricula) { %>
          <div class="invalid-feedback"><%= erroMatricula %></div>
        <% } %>
      </div>

      <div class="mb-3">
        <label for="senha" class="form-label">Senha</label>
        <input type="password" id="senha" name="senha" maxlength="16"
required class="form-control"
          placeholder="Crie uma senha" />
      </div>

      <div class="d-flex justify-content-between">
```

```
        <a href="#" onclick="window.history.back(); return false;"
class="botao-cadastro">Voltar</a>
        <button type="submit" class="botao-cadastro">Cadastrar</button>
      </div>
    </form>
  </div>
</div>

<script>
  function previewImagem(event) {
    const input = event.target;
    const img = document.getElementById('preview_foto');
    if (input.files && input.files[0]) {
      const reader = new FileReader();
      reader.onload = function (e) {
        img.src = e.target.result;
      };
      reader.readAsDataURL(input.files[0]);
    }
  }

  document.addEventListener("DOMContentLoaded", function () {
    const cpfInput = document.getElementById('CPF');
    const foneInput = document.getElementById('fone');
    const numeroInput = document.getElementById('numero');
    const cepInput = document.getElementById('cep');
    const matriculaInput = document.getElementById('matricula');

    // Máscara para CPF
    cpfInput.addEventListener('input', function (e) {
      let value = e.target.value.replace(/\D/g, '');
      if (value.length <= 11) {
        value = value
          .replace(/^(\d{3})(\d)/, '$1.$2')
          .replace(/^(\d{3})\.(\d{3})(\d)/, '$1.$2.$3')
          .replace(/^(\d{3})\.(\d{3})\.(\d{3})(\d)/, '$1.$2.$3-$4');
      }
      e.target.value = value;
    });
```

```
    // Máscara para telefone
    foneInput.addEventListener('input', function (e) {
      let value = e.target.value.replace(/\D/g, '');
      if (value.length > 11) value = value.slice(0, 11);
      value = value
        .replace(/^(\d{2})(\d)/g, '($1) $2')
        .replace(/(\d{5})(\d{1,4})$/, '$1-$2');
      e.target.value = value;
    });

    // Máscara para número da residência
    numeroInput.addEventListener('input', function (e) {
      e.target.value = e.target.value.replace(/\D/g, '');
    });

    // Máscara para CEP
    cepInput.addEventListener('input', function (e) {
      let value = e.target.value.replace(/\D/g, '');
      if (value.length > 8) value = value.slice(0, 8);
      value = value.replace(/^(\d{5})(\d)/, '$1-$2');
      e.target.value = value;
    });

    // Máscara para matrícula
    matriculaInput.addEventListener('input', function (e) {
      e.target.value = e.target.value.replace(/\D/g, '');
    });

    // Antes de enviar o formulário, remove todas as máscaras (deixa só
números)
    const form = document.querySelector('form');
    form.addEventListener('submit', function () {
      cpfInput.value = cpfInput.value.replace(/\D/g, '');
      foneInput.value = foneInput.value.replace(/\D/g, '');
      numeroInput.value = numeroInput.value.replace(/\D/g, '');
      cepInput.value = cepInput.value.replace(/\D/g, '');
      matriculaInput.value = matriculaInput.value.replace(/\D/g, '');
    });
  });
</script>
```

cadastro.ejs(do usuario/paciente):

```
<div class="d-flex justify-content-center align-items-start pt-5 pb-5" style="min-height: 100vh;">
    <div class="card shadow rounded-4 p-5 mb-5" style="width: 100%; max-width: 600px; background-color: #fff;">
      <form action="/auth/add-usuario" method="POST" enctype="multipart/form-data">
  <div class="text-center mb-4">
    <img src="/assets/logo.png" alt="Logo" style="width: 50px; height: 70px;">
    <h5 class="mt-2 logo-nome-cadastro">Criar conta</h5>

    <label for="foto_perfil">
      <img id="preview_foto" src="/assets/default-profile.jpg" alt="Foto de perfil"
        class="rounded-circle mt-3 mb-2"
        style="width: 90px; height: 90px; object-fit: cover; cursor: pointer; border: 2px solid #ccc;" />
    </label>

    <input type="file" id="foto_perfil" name="foto_perfil" accept="image/*" class="d-none" onchange="previewImagem(event)">
  </div>

  <div class="mb-3">
    <label for="nome" class="form-label">Nome</label>
    <input type="text" id="nome" name="nome" maxlength="40" required class="form-control"
      placeholder="Digite seu nome completo" value="<%= preenchido?.nome || '' %>" />
  </div>

  <div class="mb-3">
    <label for="data_nasc" class="form-label">Data de nascimento</label>
    <input type="date" id="data_nasc" name="data_nasc" required class="form-control"
      value="<%= preenchido?.data_nasc || '' %>" />
  </div>
```

```
  <div class="mb-3">
    <label for="CPF" class="form-label">CPF</label>
    <input type="text" id="CPF" name="CPF" maxlength="14" required
      class="form-control <% if (erroCPF) { %>is-invalid<% } %>"
placeholder="Digite seu CPF"
      value="<%= preenchido?.CPF || '' %>" />
    <% if (erroCPF) { %>
      <div class="invalid-feedback"><%= erroCPF %></div>
    <% } %>
  </div>

  <div class="mb-3">
    <label for="genero" class="form-label">Gênero</label>
    <select id="genero" name="genero" required class="form-select">
      <option value="" disabled <%= !preenchido?.genero ? 'selected' : ''
%>>Não selecionado</option>
      <option value="1" <%= preenchido?.genero == '1' ? 'selected' : ''
%>>Masculino</option>
      <option value="2" <%= preenchido?.genero == '2' ? 'selected' : ''
%>>Feminino</option>
      <option value="3" <%= preenchido?.genero == '3' ? 'selected' : ''
%>>Não-binário</option>
      <option value="4" <%= preenchido?.genero == '4' ? 'selected' : ''
%>>Outro</option>
    </select>
  </div>

  <div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input type="email" id="email" name="email" maxlength="30" required
class="form-control"
      placeholder="Digite seu e-mail para contato" value="<%=
preenchido?.email || '' %>" />
  </div>

  <div class="mb-3">
    <label for="fone" class="form-label">Telefone</label>
    <input type="text" id="fone" name="fone" maxlength="15"
class="form-control"
```

```
      placeholder="Digite seu telefone para contato" value="<%=
preenchido?.fone || '' %>" />
  </div>

  <hr>
  <h6 style="color:#909090">Endereço</h6>

  <div class="mb-3">
    <label for="rua" class="form-label">Rua</label>
    <input type="text" id="rua" name="rua" required class="form-control"
      placeholder="Digite a rua da sua residência" value="<%=
preenchido?.rua || '' %>" />
  </div>

  <div class="mb-3">
    <label for="numero" class="form-label">Número</label>
    <input type="text" id="numero" name="numero" required
class="form-control"
      placeholder="Digite o número da sua residência" value="<%=
preenchido?.numero || '' %>" />
  </div>

  <div class="mb-3">
    <label for="bairro" class="form-label">Bairro</label>
    <input type="text" id="bairro" name="bairro" required
class="form-control"
      placeholder="Digite o bairro da sua residência" value="<%=
preenchido?.bairro || '' %>" />
  </div>

  <div class="mb-3">
    <label for="cidade" class="form-label">Cidade</label>
    <input type="text" id="cidade" name="cidade" required
class="form-control"
      placeholder="Digite a cidade da sua residência" value="<%=
preenchido?.cidade || '' %>" />
  </div>

  <div class="mb-3">
    <label for="uf" class="form-label">Estado (UF)</label>
```

```
    <select id="uf" name="uf" required class="form-select">
      <option disabled <%= !preenchido?.uf ? 'selected' : '' %>>Não
selecionado</option>
      <% const estados =
['SC','PR','RS','SP','RJ','MG','ES','DF','GO','MT','MS','AL','BA','CE','MA
','PB','PE','PI','RN','SE','AC','AP','AM','PA','RO','RR','TO']; %>
      <% estados.forEach(estado => { %>
        <option value="<%= estado %>" <%= preenchido?.uf === estado ?
'selected' : '' %>><%= estado %></option>
      <% }); %>
    </select>
  </div>

  <!-- CEP -->
  <div class="mb-3">
    <label for="cep" class="form-label">CEP</label>
    <input type="text" id="cep" name="cep" required class="form-control"
      placeholder="Digite o número do seu CEP" value="<%= preenchido?.cep
|| '' %>" />
  </div>

  <hr>

  <div class="mb-3">
    <label for="SUS" class="form-label">Cartão SUS (CNS)</label>
    <input type="text" id="SUS" name="SUS" maxlength="15" required
class="form-control <% if (erroSUS) { %>is-invalid<% } %>"
      placeholder="Digite o número do seu cartão SUS" value="<%=
preenchido?.SUS || '' %>" />
      <% if (erroSUS) { %>
      <div class="invalid-feedback"><%= erroSUS %></div>
    <% } %>
  </div>

  <div class="mb-3">
    <label for="senha" class="form-label">Senha</label>
    <input type="password" id="senha" name="senha" maxlength="16" required
class="form-control"
      placeholder="Crie uma senha" />
  </div>
```

```
  <div class="d-flex justify-content-between">
    <a href="#" onclick="window.history.back(); return false;"
class="botao-cadastro">Voltar</a>
    <button type="submit" class="botao-cadastro">Cadastrar</button>
  </div>
</form>

    </div>
  </div>
<script>
  // Foto de perfil
  function previewImagem(event) {
    const input = event.target;
    const img = document.getElementById('preview_foto');

    if (input.files && input.files[0]) {
      const reader = new FileReader();
      reader.onload = function (e) {
        img.src = e.target.result;
      };
      reader.readAsDataURL(input.files[0]);
    }
  }

  // Máscaras de CPF e Telefone e CEP
  document.addEventListener("DOMContentLoaded", function () {
    const cpfInput = document.getElementById('CPF');
    const foneInput = document.getElementById('fone');
    const numeroInput = document.getElementById('numero');
    const cepInput = document.getElementById('cep');
    const susInput = document.getElementById('SUS');

    // Máscara para CPF
    cpfInput.addEventListener('input', function (e) {
      let value = e.target.value.replace(/\D/g, '');
      if (value.length <= 11) {
        value = value
          .replace(/^(\d{3})(\d)/, '$1.$2')
          .replace(/^(\d{3})\.(\d{3})(\d)/, '$1.$2.$3')
```

```
          .replace(/^(\d{3})\.(\d{3})\.(\d{3})(\d)/, '$1.$2.$3-$4');
      }
      e.target.value = value;
    });

    // Máscara para telefone
    foneInput.addEventListener('input', function (e) {
      let value = e.target.value.replace(/\D/g, '');
      if (value.length > 11) value = value.slice(0, 11);

      value = value
        .replace(/^(\d{2})(\d)/g, '($1) $2')
        .replace(/(\d{5})(\d{1,4})$/, '$1-$2');

      e.target.value = value;
    });


    // Máscara para numero da residencia -> /\D/g -> remove caracteres
globalmente e substitui por ''
    numeroInput.addEventListener('input', function (e) {
      e.target.value = e.target.value.replace(/\D/g, '');
    });


    // Máscara para CEP
     cepInput.addEventListener('input', function (e) {
       let value = e.target.value.replace(/\D/g, '');
       if (value.length > 8) value = value.slice(0, 8);
       value = value.replace(/^(\d{5})(\d)/, '$1-$2');
       e.target.value = value;
     });

    // Remove não numéricos do SUS
      susInput.addEventListener('input', function (e) {
        e.target.value = e.target.value.replace(/\D/g, '');
      });

    // Antes de enviar o formulário, remove as máscaras
    const form = document.querySelector('form');
```

```
    form.addEventListener('submit', function (e) {
      // Remove qualquer caractere não numérico antes de enviar
      cpfInput.value = cpfInput.value.replace(/\D/g, '');
      foneInput.value = foneInput.value.replace(/\D/g, '');
      numeroInput.value = numeroInput.value.replace(/\D/g,'');
      cepInput.value = cepInput.value.replace(/\D/g, '');
      susInput.value = susInput.value.replace(/\D/g,'');

    });
  });
</script>
```