Faculdade de Informática e Administração Paulista
Análise e Desenvolvimento de Sistemas

Desenvolvedores:

- Eduardo Finardi | RM: 98624
- Miguel Santos | RM: 551640
- Murilo Munari | RM: 94164
- Felipe Morais | RM: 551463
- João Cardoso | RM: 552078

PROJETO BIKE CHECK

Sistema de Análise online de bicicletas

São Paulo
2023

Descrição

O projeto BikeCheck é um sistema especializado em análise de fotos de bicicletas, dentro de um intervalo de preço de 1000 a 100.000 reais que deve ser entregue à empresa Porto no final da Quarta Sprint, determinada para a data 13/11/2023.

Como o sistema é necessariamente especializado apenas na parte de análise, validação e feedback das fotos de bicicleta, ele não engloba parte de login/cadastro de usuários (Porto já possui esse sistema) e nem parte contratual (que envolve questão de preço, plano de assinatura, entre outras burocracias), portanto o sistema vai envolver apenas o solicitado, que é a análise.

O funcionamento front-end e os passos a serem seguidos pelo sistema serão de extrema simplicidade com uma interface agradável, sendo o complexo apenas focado no back-end, na parte da do Machine Learning. Começando pelas 3 funcionalidades principais do sistema, sendo elas: selecionar se ele vai realizar o primeiro seguro, se vai renovar o seguro, acionar o seguro em caso de acidente.

Vamos por partes, caso seja escolhida a primeira opção, respectivamente das 3 apresentadas, o sistema encaminhará ele para o setor de primeiro seguro, mostrando uma primeira tela que solicitará os dados da bicicleta, o usuário deve informar obrigatoriamente as principais peças da bicicleta (rodas, quadro, guidão, freios, selim, coroa e suspensão), sendo o resto opcional, aceitando valores “null”. Após esse processo, será solicitado o valor pago pela bicicleta, e fotos das notas fiscais para validação. Dando sequência a todos esses processos, caso seja realizado com sucesso, o sistema armazenará todas as fotos, enviará para um banco de dados e nossa Inteligência Artificial fará a devida comparação das fotos das peças enviadas pelo cliente com fotos do sistema (fotos pré-existentes da peça armazenada no banco) e a partir da comparação e semelhança das fotos, dará se a bicicleta é como ela está declarada. Caso a validação seja efetuada com sucesso e a bicicleta seja exatamente como ela está declarada, o cliente será encaminhado para área contratual dos serviços da Porto, parte que não faz parte de nosso sistema, assim se encerrando a primeira opção.

Já se a segunda opção for escolhida o processo será muito parecido com o primeiro, mudando apenas os passos iniciais a serem seguidos pelo sistema. Portanto, caso seja a segunda opção a selecionada, a de renovação de seguro, o cliente primeiramente declarará se houveram alterações nas peças da bicicleta, pois mudanças na bicicleta alteram o preço dela, por consequência mudando o valor do seguro, pois a bicicleta adquiriu um valor agregado maior. Se não houverem alterações, não tem peças a serem declaradas (pois já foram declaradas anterioramente), então levará o cliente direito para solicitação de fotos, depois de serem devidamente recebida as fotos, os passos a serem seguidos são os mesmo do descrito previamente no parágrafo anterior, a partir da parte de armazenamento de fotos do sistema dentro do banco de dados.

E para finalizar, em caso de acionamento do seguro em caso de acidente segue novamente, um fluxo diferente em seu começo, mas tendo seu final igual ao dos dois casos descritos acima, a partir da parte de armazenamento de fotos do sistema dentro do banco de dados. Deste modo, sendo escolhida a terceira opção,
O sistema solicitará uma descrição do acidente (peças danificadas, onde ocorreu, como aconteceu, horário do ocorrido, se foi feito por terceiros ou pelo proprietário da bicicleta), após a coleta da descrição, ela será enviada para análise. Na sequência da aprovação da análise será solicitado fotos do incidente para análise de danos e veracidade, concluindo a análise e sendo verificado como verídico o caso, o cliente será encaminhado para funcionários da Porto para dar sequência no processo de reposição das peças.

Detalhes adicionais importantes, qualquer passo que não for bem sucedido (fugindo do fluxo principal), durante todos os 3 processos apresentados, basicamente todos darão a opção de recomeçar os casos, finalizá-los ou acionar assistência de funcionários da Porto.

As principais funcionalidades do sistema já foram apresentadas, agora vamos para as tecnologias utilizadas para os desenvolvimentos de cada parte do sistema. Para front-end está sendo feito em Next, desenvolvido pela plataforma Vscode. Para back-end está sendo utilizado Python, desenvolvido pela plataforma Vscode, e Java (focado na parte de POO), sendo desenvolvida pela plataforma IntelliJ. Como parte do back-end, a inteligência artificial está sendo desenvolvida em Python se utilizando de bibliotecas específicas, como:TensorFlow, requests, openCV, Flask, Keras, cx_Oracle. O local de desenvolvimento da Inteligência Artificial foi feito pelo Google Collab, como plataforma de desenvolvimento, e parte do código proveniente da plataforma da Teachable Machine. O Banco de Dados está sendo desenvolvido em SQL se utilizando dos servidores da Oracle e da plataforma de desenvolvimento SQL Developer. Já na arquitetura de Software foi utilizado Astah, Trello e Word para a documentação e organização.

O treinamento da IA está sendo realizado da seguinte maneira, serão pegas 300 fotos no total, englobando cada umas das peças principais da bicicleta, que seriam: rodas, quadro, guidão, freios, selim, cassete, coroa e suspensão. Cada uma dessas peças são classes ou subclasses, dentro do projeto, sendo o restante das peças da bicicleta (que totalizam 22), são atributos das peças principais. Dentro dessas fotos foram separadas labels de leitura para a IA conseguir identificar peça a peça, a seguir dois exemplos, um de roda e outro de quadro:

As labels, respectivamente em preto e vermelho, serão as áreas de leitura da IA. Com a coleta de diferentes fotos e ângulos dos 3 diferentes modelos e marcas mais comuns no mercado de seguros. Serão apresentados aos usuários pelo menos 3 maneiras de tirar as fotos para evitar erros por parte do cliente, impedindo nossa IA de agir. As fotos devem ser retiradas das principais partes da bicicleta, serão solicitadas respectivamente uma foto da bicicleta geral (pela lateral), duas fotos de cada roda (pela lateral também), 3 fotos do quadro (peça intermediária que liga todas as peças da bicicleta, sendo o ponto principal da análise), uma do selim e do guidão (com ênfase na marca, caso ela esteja exposta). A partir dessa quantidade de fotos, sem estarem tremidas e nos ângulo solicitados, nossa inteligência será capaz de identificar cada peça que compõe a bicicleta e suas respectivas marcas, assim validando com as declaradas e dando sequência no processo. O programa consumirá uma API própria, que será o site onde o cliente fará o upload das fotos, recebendo no mesmo site a resposta da análise, se o programa reconhecer a peça, ele fornecerá o nome da classe, fará a devida comparação com a peça declarada e trará se ambos são compatíveis.

Prints Java, InteliJ:

```
2 usages  murilomunari
@Override
public List<Bicicleta> findAll() {
    List<Bicicleta> list = new ArrayList<>();
    Connection con = factory.getConnection();
    ResultSet rs = null;
    Statement st = null;
    try {
        String sql = "SELECT * FROM TB_BICICLETA";
        st = con.createStatement();
        rs = st.executeQuery(sql);
        if (rs.isBeforeFirst()){
            ClienteRepository clienteRepo = ClienteRepository.build();
            while (rs.next()){
                Long id = rs.getLong( columnLabel: "ID_BICICLETA");
                String nome = rs.getString( columnLabel: "TIPO_BICICLETA");
                Long idCliente = rs.getLong( columnLabel: "DONO");
                Cliente dono = clienteRepo.findById(idCliente);
                list.add(new Bicicleta(id, nome, dono));
            }
        }

    }catch (SQLException e){
        System.err.println("Não foi possivel consultar a bicicleta" + e.getMessage());
    }finally {
        fecharObjetos(rs,st,con);
    }
    return list;
}
```

```java
@Override
public List<Bicicleta> findAll() {
    List<Bicicleta> list = new ArrayList<>();
    Connection con = factory.getConnection();
    ResultSet rs = null;
    Statement st = null;
    try {
        String sql = "SELECT * FROM TB_BICICLETA";
        st = con.createStatement();
        rs = st.executeQuery(sql);
        if (rs.isBeforeFirst()){
            ClienteRepository clienteRepo = ClienteRepository.build();
            while (rs.next()){
                Long id = rs.getLong( columnLabel: "ID_BICICLETA");
                String nome = rs.getString( columnLabel: "TIPO_BICICLETA");
                Long idCliente = rs.getLong( columnLabel: "DONO");
                Cliente dono = clienteRepo.findById(idCliente);
                list.add(new Bicicleta(id, nome, dono));
            }
        }

    }catch (SQLException e){
        System.err.println("Não foi possivel consultar a bicicleta" + e.getMessage());
    }finally {
        fecharObjetos(rs,st,con);
    }
    return list;
```

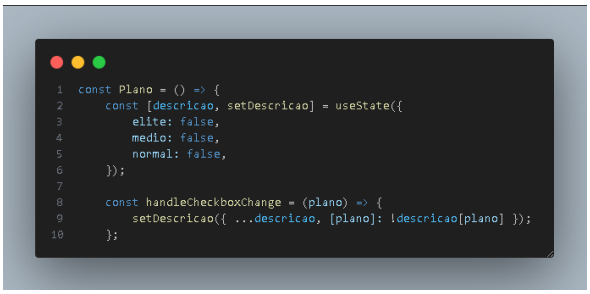
```java
2 usages  murilomunari
@Override
public Bicicleta persist(Bicicleta bicicleta) {
    var sql = "INSERT INTO TB_BICICLETA (ID_BICICLETA, TIPO_BICICLETA, DONO) VALUES (0,?,?)";

    Connection conn = factory.getConnection();
    PreparedStatement ps = null;
    try {
        ps = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
        ps.setString( parameterIndex: 1, bicicleta.getTipo_bicicleta());

        ps.executeUpdate();

        final ResultSet rs = ps.getGeneratedKeys();

        if (rs.next()){
            final Long id = rs.getLong( columnIndex: 1);
            bicicleta.setId(id);
        }
    }catch (SQLException e){
        System.err.println("Não foi posivel salvar a bicicleta no banco de dados: " + e.getMessage());
    }finally {
        fecharObjetos( rs: null, ps, conn);
    }
    return bicicleta;
}
```

```java
2 usages  murilomunari +1
@Override
public Cliente persist(Cliente cliente) {
    var sql = "INSERT INTO TB_CLIENTE (ID_CLIENTE, ENDERECO, NM_CLIENTE, NR_CPF) VALUES (0,?,?,?)";

    Connection con = factory.getConnection();
    PreparedStatement ps = null;

    try {
        ps = con.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);

        ps.setString( parameterIndex: 1,cliente.getNome());
        ps.setString( parameterIndex: 2,cliente.getEndereco());
        ps.setString( parameterIndex: 3, cliente.getCpf());

        ps.executeUpdate();

        final  ResultSet rs = ps.getGeneratedKeys();

        if (rs.next()){
            final Long id = rs.getLong( columnIndex: 1);
            cliente.setId(id);
        }
    }catch (SQLException e){
        System.err.println("Não foi possivel inserir os dados!\n" + e.getMessage());
    }finally {
        fecharObjetos( rs: null, ps, con);
    }
    return cliente;
}
```

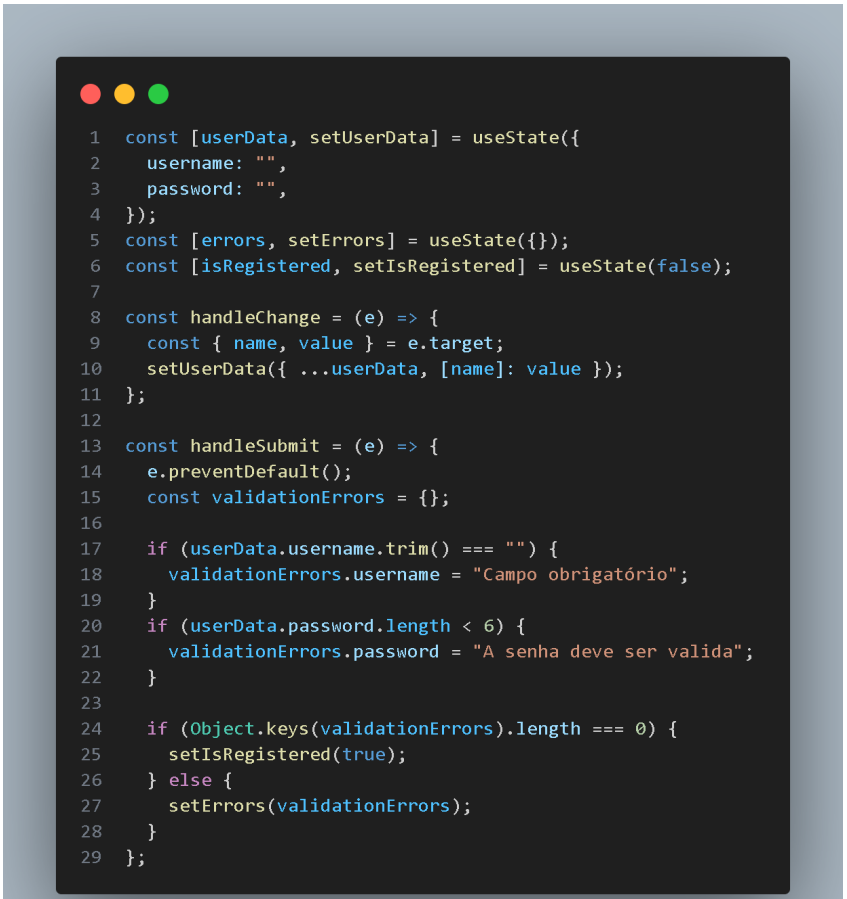
```java
2 usages  murilomunari +1
@Override
public List<Cliente> findAll() {
    List<Cliente> list = new ArrayList<>();
    Connection con = factory.getConnection();
    ResultSet rs = null;
    Statement st = null;
    try {
        String sql = "SELECT * FROM TB_CLIENTE";
        st = con.createStatement();
        rs = st.executeQuery(sql);
        if (rs.isBeforeFirst()){
            while(rs.next()){
                Long id = rs.getLong( columnLabel: "ID_CLIENTE");
                String nome = rs.getString( columnLabel: "NM_CLIENTE");
                String endereco = rs.getString( columnLabel: "ENDERECO");
                String cpf = rs.getString( columnLabel: "NR_CPF");
                list.add(new Cliente(id, nome, endereco, cpf));
            }
        }
    }catch (SQLException e){
        System.err.println("Não foi possivel consultar os dados!\n" + e.getMessage());
    }finally {
        fecharObjetos(rs,st,con);
    }
    return list;
}
```

```java
4 usages  murilomunari +1
@Override
public Cliente findById(Long id) {
    Cliente cliente = null;
    var sql  = "SELECT * FROM TB_CLIENTE where ID_CLIENTE = ?";
    Connection con = factory.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        ps = con.prepareStatement(sql);
        ps.setLong( parameterIndex: 1, id);
        rs = ps.executeQuery();
        if (rs.isBeforeFirst()){
            while(rs.next()){
                String nome = rs.getString( columnLabel: "NM_CLIENTE");
                String endereco = rs.getString( columnLabel: "ENDERECO");
                String cpf = rs.getString( columnLabel: "NR_CPF");
                cliente = new Cliente(id, nome, endereco, cpf);
            }
            }else {
            System.out.println("Dados não encontrados com o id: " + id);
        }
    }catch (SQLException e){
        System.err.println("Não foi possivel consultar os dados!\n" + e.getMessage());
    }finally {
        fecharObjetos(rs, ps, con);
    }
    return cliente;
}
```

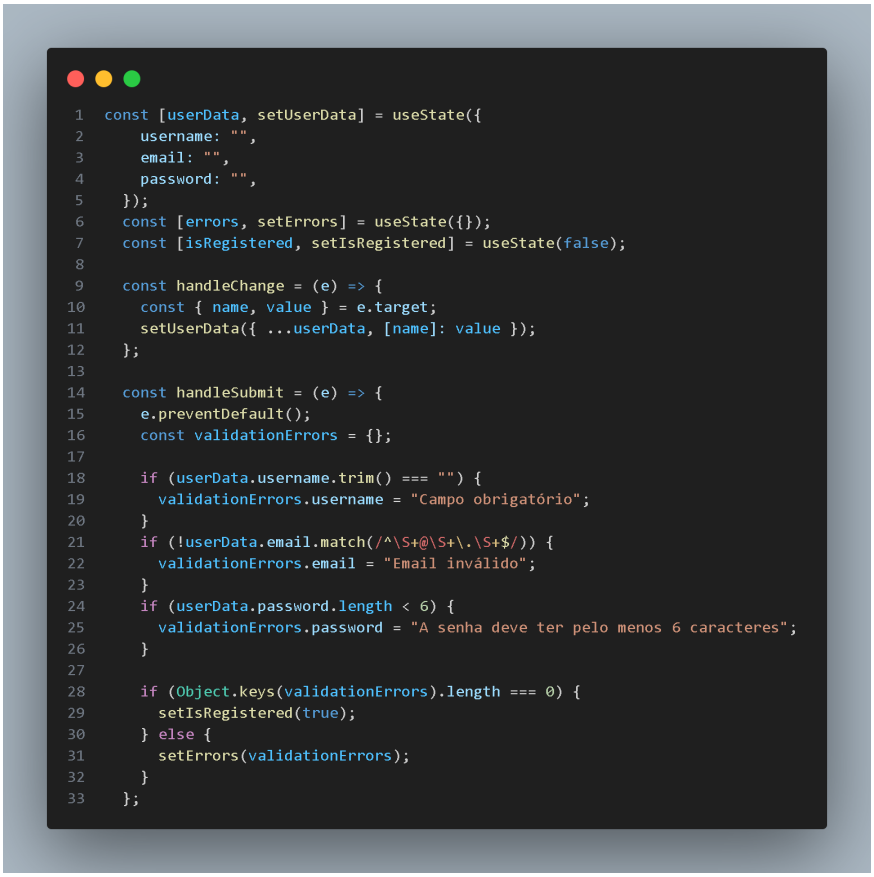
Prints RWD(JavaScript), VScode:



```javascript
const Plano = () => {
    const [descricao, setDescricao] = useState({
        elite: false,
        medio: false,
        normal: false,
    });

    const handleCheckboxChange = (plano) => {
        setDescricao({ ...descricao, [plano]: !descricao[plano] });
    };
```



```
const [userData, setUserData] = useState({
  username: "",
  password: "",
});
const [errors, setErrors] = useState({});
const [isRegistered, setIsRegistered] = useState(false);

const handleChange = (e) => {
  const { name, value } = e.target;
  setUserData({ ...userData, [name]: value });
};

const handleSubmit = (e) => {
  e.preventDefault();
  const validationErrors = {};

  if (userData.username.trim() === "") {
    validationErrors.username = "Campo obrigatório";
  }
  if (userData.password.length < 6) {
    validationErrors.password = "A senha deve ser valida";
  }

  if (Object.keys(validationErrors).length === 0) {
    setIsRegistered(true);
  } else {
    setErrors(validationErrors);
  }
};
```



```
const [userData, setUserData] = useState({
    username: "",
    email: "",
    password: "",
  });
  const [errors, setErrors] = useState({});
  const [isRegistered, setIsRegistered] = useState(false);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setUserData({ ...userData, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    const validationErrors = {};

    if (userData.username.trim() === "") {
      validationErrors.username = "Campo obrigatório";
    }
    if (!userData.email.match(/^\S+@\S+\.\S+$/)) {
      validationErrors.email = "Email inválido";
    }
    if (userData.password.length < 6) {
      validationErrors.password = "A senha deve ter pelo menos 6 caracteres";
    }

    if (Object.keys(validationErrors).length === 0) {
      setIsRegistered(true);
    } else {
      setErrors(validationErrors);
    }
  };
```

Prints Banco de Dados:

```
DECLARE
  cliente_id NUMBER;
BEGIN
  -- Inserção na Tabela CLIENTE
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) VALUES (seq_CLIENTE.nextval, TO_DATE('12/03/2005', 'DD/MM/YYYY'),'jaofs12.03@gmail.com', '11984399932', 'joao', '08040460', '645375386');
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) Values (seq_CLIENTE.nextval, TO_DATE('13/02/2001', 'DD/MM/YYYY'),'blats12.03@gmail.com','1198935932','felipe','0365860','698775386');
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) Values (seq_CLIENTE.nextval, TO_DATE('11/08/2002', 'DD/MM/YYYY'),'edulats12.03@gmail.com','1197605932','eduardo','09031260','01475386');
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) Values (seq_CLIENTE.nextval, TO_DATE('09/07/2003', 'DD/MM/YYYY'),'miguel12.03@gmail.com','11960305932','miguel','08732260','2215386');
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) Values (seq_CLIENTE.nextval, TO_DATE('15/06/1997', 'DD/MM/YYYY'),'murilaol12.03@gmail.com','1190405932','murilo','08532332','2798467348');
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) Values (seq_CLIENTE.nextval, TO_DATE('10/04/1999', 'DD/MM/YYYY'),'lucaoll12.03@gmail.com','11933405932','lucas','08738630','2073624878');
  INSERT INTO CLIENTE (ID_CLIENTE, DT_NASCIMENTO, E_MAIL, NR_TELEFONE, NM_NOME, CEP, CPF) Values (seq_CLIENTE.nextval, TO_DATE('02/02/1992', 'DD/MM/YYYY'),'fredericol12.03@gmail.com','1197653932','frederico','08756632','2002363');


  -- Obtendo o valor atual da sequência seq_CLIENTE
  SELECT seq_CLIENTE.currval INTO cliente_id FROM DUAL;

  -- Início do bloco PL/SQL aninhado
  DECLARE
    id_bicicleta NUMBER;
  BEGIN
    -- Inserções nas tabelas BICICLETA, GUIDAO, SELIM, QUADRO, RODAS

    INSERT INTO BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval, 'caloi2022', 1500.50, cliente_id);
    INSERT INTO BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval ,'caloi2022', 1500.50, cliente_id);
    insert into BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval ,'oggispeed', 1000.20, cliente_id);
    insert into BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval ,'sensetracker', 1100.10, cliente_id);
    insert into BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval ,'oggi2012', 1300.12, cliente_id);
    insert into BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval ,'caloi2013', 1250.11, cliente_id);
    insert into BICICLETA (ID_BICICLETA, MODELO_DA_BICICLETA, NOTA_FISCAL, ID_CLIENTE) VALUES (seq_BICICLETA.nextval ,'caloi2011', 1250.11, cliente_id);
    -- Inserções GUIDAO
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval, 'mtb', 'manopladeborrachamtb', 'maneteshipanotp', 'alavancax-TimeTr', seq_BICICLETA.currval);
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval,'absolute','manoplaDeFerro','maneteGTA','alavancaGT53X', seq_BICICLETA.currval);
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval,'alfameq','manoplaGTSBorracha','maneteTrial', 'lavancaPaco', seq_BICICLETA.currval);
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval,'avance','manoplaAbsoluteNBR','maneteLogan','alavancaGTA',seq_BICICLETA.currval);
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval,'Calypso','manoplaBefit','maneteCicloHouse', 'alavancaShimano', seq_BICICLETA.currval);
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval,'caicara','manoplaJWS','maneteTSW','alavancaPeak',seq_BICICLETA.currval);
    INSERT INTO GUIDAO (ID_GUIDAO, MARCA_GUIDAO, MANOPLAS, MANETE_DE_FREIO, ALAVANCA_DE_CAMBIO, ID_BICICLETA) VALUES (seq_GUIDAO.nextval,'AOLIXIN','manoplaTMBYX','maneteIsapa','alavancaMagideal',seq_BICICLETA.currval);
    -- Inserções Selim
    INSERT INTO SELIM (ID_SELIM, MARCA_SELIM, CANOTE, MESA, ABRACADEIRA, ID_BICICLETA) VALUES (seq_SELIM.nextval, 'selimpowercomp', 'canotedeselimAbsolute', 'mesaMTB', 'abracadeiraAllen', seq_BICICLETA.currval);
    INSERT INTO SELIM (ID_SELIM, MARCA_SELIM, CANOTE, MESA, ABRACADEIRA, ID_BICICLETA) VALUES (seq_SELIM.nextval, 'selimMTB','canotedeselimGTA','mesaAbsolute', 'abracadeiraAbsolute', seq_BICICLETA.currval);
    INSERT INTO SELIM (ID_SELIM, MARCA_SELIM, CANOTE, MESA, ABRACADEIRA, ID_BICICLETA) VALUES (seq_SELIM.nextval,'selimAifeit','canotedeselimGTS','mesaPromax','abracadeiraMuqzi',seq_BICICLETA.currval);
    INSERT INTO SELIM (ID_SELIM, MARCA_SELIM, CANOTE, MESA, ABRACADEIRA, ID_BICICLETA) VALUES (seq_SELIM.nextval,'selimGios','canotedeselimWild','mesaFmfxtr', 'abracadeiraNeroi',seq_BICICLETA.currval);
    INSERT INTO SELIM (ID_SELIM, MARCA_SELIM, CANOTE, MESA, ABRACADEIRA, ID_BICICLETA) VALUES (seq_SELIM.nextval, 'selimitx','canotedeselimGlossy','mesaUno','abracadeiraAloyy',seq_BICICLETA.currval);
    INSERT INTO SELIM (ID_SELIM, MARCA_SELIM, CANOTE, MESA, ABRACADEIRA, ID_BICICLETA) VALUES (seq_SELIM.nextval, 'selimGta','canotedeselimSenTec','mesaBrave','abracadeiraToken',seq_BICICLETA.currval);
      -- Inserções Quadros
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval, 'quadroPowercomp', 'pedalShimano', 'pedivelaGTS', 'caixaDeDirecaoAllen', 'garfoDeFerro', seq_BICIC
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval,'quadroAbsolute','pedalVenzo','pedivelaGTA','aixaDeDirecaoAbsolute', 'garfoAbsolute',seq_BICICLETA.
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval,'quadroGTI','pedalWelgo','pedivelaIXF','caixaDeDirecaoNeco', 'garfoAllen',seq_BICICLETA.currval);
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval,'quadroGios','pedalRoyal','pedivelaShimano','caixaDeDirecaoCly','garfoAlloy' ,seq_BICICLETA.currval
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval,'quadroGantech','pedalCiclo','pedivelaSugino','caixaDeDirecaoPaco','garfoGTS' ,seq_BICICLETA.currva
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval,'quadroForss','pedalMetal','pedivelaGTS','caixaDeDirecaoMega','garfoPaco', seq_BICICLETA.currval);
    INSERT INTO QUADRO (ID_QUADRO, MARCA_QUADRO, PEDAL, PEDIVELA, CAIXA_DE_DIRECAO, GARFO, ID_BICICLETA) VALUES (seq_QUADRO.nextval,'quadroKylin','pedalCrank','pedivelaCross','caixaDeDirecaoYamada','garfoShimano' ,seq_BICICLETA.cur
    -- Inserções Rodas
    INSERT INTO RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval, 'rodasAbsolute', 'freioShimano', 'cambioTraseiroAbsolute', 'cam
    insert into RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval,'rodasDt','freioTrekt','cambioTraseiroShimano','cambioDianteiroS
    insert into RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval,'rodasSentec','freioJagwire','cambioTraseiroSun','cambioDianteir
    insert into RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval,'rodasVzan','freioHigh','cambioTraseiroRace','cambioDianteiroRac
    insert into RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval,'rodasKfb','freioOne','cambioTraseiroWg','cambioDianteiroWg','ca
    insert into RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval,'rodasAlexin','freioAligattor','cambioTraseiroSram','cambioDiant
    insert into RODAS (ID_RODAS, MARCA_RODAS, FREIO, CAMBIO_TRASEIRO, CAMBIO_DIANTEIRO, CASSETE, CORRENTE, SUSPENSAO_TRASEIRA, ID_BICICLETA) VALUES (seq_RODAS.nextval,'rodasShimano','freioAbsolute','cambioTraseiroTiagra','cambioDia

    -- Obtendo o valor atual da sequência seq_BICICLETA
    SELECT seq_BICICLETA.currval INTO id_bicicleta FROM DUAL;


  END; -- Fim do bloco PL/SQL aninhado

  -- Restante do bloco externo, se houver

END;
/
```