Nome da Empresa: Evonin

Ramo da Empresa: Varejo

Link Github: clique aqui

Equipe Responsável:

Jean Lucas Oliveira de Aguiar RA: 2222101720
Vinicius Meneses da Silva RA: 2222103940
Eric Zonato lima RA: 2222103734
Leonardo Nicolas Barbosa RA: 2222104964
Anderson Fernandes Filho RA: 2222102272

Curso: Análise e desenvolvimento de sistemas

Turma: 40

Semestre: 5

Ano: 2024

Sumário

1. Escopo do projeto:

O presente documento detalha o escopo do projeto proposto pela empresa Evonin.

A Evnonin é uma empresa de varejo especializada na venda de produtos tecnológicos através da internet. a Evnonin destaca-se pela sua interface de usuário intuitiva e navegação facilitada. Nosso propósito é simplificar o processo de aquisição de produtos tecnológicos, permitindo que os clientes encontrem e comprem os itens desejados de maneira eficiente e sem complicações.

2. Serviços Oferecidos:

- Loja Online Intuitiva
- Entrega Rápida e Segura
- Suporte ao Cliente
- Programas de Fidelidade e Promoções
- Serviços de Pagamento Flexíveis
- Consultoria e Recomendações Personalizadas
- Política de Devolução e Troca Fácil

3. Estruturação Interna da Empresa

3.1 Aprendizado de máquina

Para acessar o repositório do GitHub clique aqui

3.1.1 - Entrega 1: Exploração de Dados e Pré-processamento

Coleta de dados relevantes para o negócio proposto pela empresa:

```
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score,
recall_score
from sklearn.model_selection import train_test_split,
GridSearchCV, cross_val_score
from sklearn.ensemble import RandomForestClassifier

# Carregar dados de cada planilha em um DataFrame
produtos_df = pd.read_excel('produtos.xlsx')
clientes_df = pd.read_excel('clientes.xlsx')
pedidos_df = pd.read_excel('pedidos.xlsx')
itens_pedido_df = pd.read_excel('itens_pedido.xlsx')
estoque_df = pd.read_excel('estoque.xlsx')
categorias_df = pd.read_excel('categorias.xlsx')
fornecedores_df = pd.read_excel('fornecedores.xlsx')
```

Limpeza e pré-processamento dos dados:

```
# Tratar valores ausentes, outliers e dados inconsistentes
produtos_df = produtos_df.dropna()  # Remover linhas com
valores ausentes

# Confirma a remoção de linhas com valores ausentes
print("Produtos após remover valores ausentes:")
print(produtos_df)

# Padroniza formatos e unidades
produtos_df['preço'] = produtos_df['preço'].astype(float)  #

# Confirma a conversão de formato
print("\nFormato dos preços após conversão:")
print(produtos_df['preço'].dtype)
```

Verificar a matriz confusão:

```
# Classificação dos produtos com base no preço
produtos_df['classificação'] =
```

```
produtos_df['preço'].apply(lambda x: 1 if x > 50 else 0)

# Valores verdadeiros (preços reais) e valores previstos
(classificação baseada no preço)
y_true = produtos_df['classificação']
y_pred = produtos_df['classificação']

# Calcula a matriz de confusão
matriz_confusao = confusion_matrix(y_true, y_pred)

# Exibe a matriz de confusão
print("Matriz de Confusão:")
print(matriz_confusao)
```

3.1.2 - Entrega 2: Implementação de Modelos de Aprendizado de Máquina

Escolha de algoritmos de ML adequados ao problema:

```
# Dividir os dados em conjunto de treinamento e teste
X = produtos_df[['preço']]  # Features
y = produtos_df['classificação']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Escolher algoritmo de classificação
modelo = RandomForestClassifier()
```

Desenvolva e treine os modelos selecionados.

```
# Treinar o modelo
modelo.fit(X_train, y_train)
```

Avaliação da performance dos modelos com métricas apropriadas:

```
# Avaliar a precisão do modelo
precisao = accuracy_score(y_test, y_pred)
print("Precisão do modelo:", precisao)

# Identifica corretamente todos os exemplos positivos.
recall = recall_score(y_test, y_pred)
print("Recall do modelo:", recall)
```

3.1.3 - Entrega 3: Otimização e Validação do Modelo

Otimização dos hiperparâmetros dos modelos para melhorar a performance:

```
# Definir os parâmetros para otimização
parametros = {'n_estimators': [50, 100, 150], 'max_depth':
[None, 10, 20]}

# Criar o objeto GridSearchCV
grid_search = GridSearchCV(RandomForestClassifier(),
parametros, cv=3)

# Executar a busca em grade
grid_search.fit(X_train, y_train)

# Melhores hiperparâmetros encontrados
print("Melhores hiperparâmetros encontrados:")
print(grid_search.best_params_)

# Avaliar o modelo com os melhores hiperparâmetros
melhor_modelo = grid_search.best_estimator_
y_pred = melhor_modelo.predict(X_test)
```

Validação cruzada para verificar a robustez do modelo:

```
# Realizar validação cruzada
pontuacoes = cross_val_score(melhor_modelo, X, y, cv=3)

# Exibir as pontuações de validação cruzada
print("Pontuações de validação cruzada:", pontuacoes)

# Calcular a média das pontuações
media_pontuacoes = pontuacoes.mean()
print("Média das pontuações de validação cruzada:",
media_pontuacoes)
```

Elabore documentação detalhada sobre o processo, parâmetros e resultados.

Etapas do Processo:

1. Exploração de Dados e Pré-processamento:

Coleta de Dados: Os dados foram coletados de várias fontes, incluindo planilhas de produtos, clientes, pedidos, itens de pedido, estoque, categorias e fornecedores.

Limpeza e Pré-processamento: Foram identificados e removidos valores ausentes nas planilhas de produtos.

A coluna de preço foi convertida para o tipo float para padronizar o formato dos dados.

2. Implementação de Modelos de Aprendizado de Máquina:

Escolha de Algoritmos:

Optou-se por utilizar o algoritmo RandomForestClassifier devido à sua capacidade de lidar com problemas de classificação e sua robustez em relação a overfitting.

Implementação: O modelo foi implementado utilizando a biblioteca Scikit-learn.

Os dados de treinamento foram alimentados no modelo RandomForestClassifier.

3. Otimização e Validação do Modelo:

Otimização de Hiperparâmetros: Foi realizada uma busca em grade utilizando GridSearchCV para encontrar os melhores hiperparâmetros para o modelo.

Os hiperparâmetros ajustados incluem 'n_estimators' (número de árvores na floresta) e 'max_depth' (profundidade máxima das árvores).

Validação Cruzada: A validação cruzada foi realizada utilizando a função cross_val_score com 3 divisões.

Os resultados obtidos indicam a precisão do modelo em diferentes conjuntos de dados.

Parâmetros do Modelo:

Os melhores hiperparâmetros encontrados foram 'n_estimators': 50 e 'max_depth': None.

Outros parâmetros relevantes para o modelo incluem o critério de divisão, o número máximo de features consideradas em cada divisão, etc.

Métricas de Avaliação: As métricas utilizadas para avaliar o desempenho do modelo incluem precisão e recall.

A precisão mede a proporção de verdadeiros positivos em relação a todos os exemplos classificados como positivos.

O recall mede a proporção de verdadeiros positivos em relação a todos os exemplos positivos reais.

Resultados:

O modelo apresentou uma precisão de 1.0 e recall de 1.0, indicando um desempenho perfeito nos dados de teste.

As pontuações de validação cruzada também mostraram uma precisão consistente de 1.0 em diferentes conjuntos de dados, indicando a robustez do modelo.

Esta documentação fornece uma visão abrangente do processo de construção e treinamento do modelo, bem como dos resultados obtidos, permitindo uma compreensão clara do modelo desenvolvido e sua capacidade de classificar produtos com base no preço.

3.2 Ciência de Dados (Python e Estatística):

Para acessar o repositório do GitHub clique aqui

3.2.1 - Entrega 1: Análise Descritiva dos Dados

Utilização de técnicas estatísticas básicas para descrever os dados:

```
#Estatísticas descritivas
media = produtos_df['preço'].mean()
mediana = produtos_df['preço'].median()
desvio_padrao = produtos_df['preço'].std()

print(f'Média: {media}, Mediana: {mediana}, Desvio Padrão:
{desvio_padrao}')
```

Visualização de dados utilizando bibliotecas como Matplotlib e Seaborn:

```
# Gráfico de dispersão entre preço e quantidade de itens
pedidos
plt.figure(figsize=(10, 6))
sns.scatterplot(x='preço_unitário', y='quantidade',
data=itens_pedido_df)
plt.title('Relação entre Preço e Quantidade de Itens Pedidos')
plt.xlabel('Preço Unitario')
plt.ylabel('Quantidade')
plt.show()
```

Identificação de padrões e tendências nos dados:

```
# Identificação de padrões nos dados
# Exemplo: Contagem de produtos por categoria
tendencia_categoria = produtos_df['nome'].value_counts()
print('Contagem de Produtos por Categoria:')
print(tendencia_categoria)
```

3.2.2 - Entrega 2: Modelagem Estatística

Aplicação de técnicas estatísticas avançadas para modelagem dos dados:

```
# Calculando o preço total do pedido (variável dependente)
itens_pedido_df['preco_total'] = itens_pedido_df['quantidade']
* itens_pedido_df['preço_unitário']

# Separando as variáveis independentes (X) e a variável
dependente (y)
X = itens_pedido_df[['quantidade', 'preço_unitário']]  #
Variáveis independentes
y = itens_pedido_df['preco_total']  # Variável dependente

# Adicionando uma constante (intercepto) ao modelo
X = sm.add_constant(X)

# Ajustando o modelo de regressão linear
modelo = sm.OLS(y, X)
resultados = modelo.fit()

# Exibindo um resumo dos resultados da regressão
print(resultados.summary())
```

Uso de ferramentas como regressão linear, classificação etc.:

```
# Calculando o preço total do pedido (variável dependente)
itens_pedido_df['preco_total'] = itens_pedido_df['quantidade']
* itens_pedido_df['preço_unitário']

# Separando as variáveis independentes (X) e a variável
dependente (y)
X = itens_pedido_df[['quantidade', 'preço_unitário']]  #
Variáveis independentes
y = itens_pedido_df['preco_total']  # Variável dependente

# Adicionando uma constante (intercepto) ao modelo
X = sm.add_constant(X)

# Ajustando o modelo de regressão linear
modelo = sm.OLS(y, X)
resultados = modelo.fit()

# Exibindo um resumo dos resultados da regressão
print(resultados.summary())

# Inicializar o modelo de regressão linear
modelo = LinearRegression()
```

Avaliação da adequação dos modelos estatísticos aos dados:

```
# Calculando o preço total do pedido (variável dependente)
itens_pedido_df['preco_total'] = itens_pedido_df['quantidade']
```

```
* itens_pedido_df['preço_unitário']

# Separando as variáveis independentes (X) e a variável
dependente (y)
X = itens_pedido_df[['quantidade', 'preço_unitário']]  #
Variáveis independentes
y = itens_pedido_df['preco_total']  # Variável dependente

# Adicionando uma constante (intercepto) ao modelo
X = sm.add_constant(X)

# Ajustando o modelo de regressão linear
modelo = sm.OLS(y, X)
resultados = modelo.fit()

# Exibindo um resumo dos resultados da regressão
print(resultados.summary())

# Inicializar o modelo de regressão linear
modelo = LinearRegression()

# Avaliar modelo
R2 = resultados.rsquared
print(f'R-squared: {R2}')
```

Implementação de modelos preditivos utilizando Python:

```
# Definir as características dos novos itens do pedido
X_treino = itens_pedido_df[['quantidade', 'preço_unitário']]
y_treino = itens_pedido_df['id_produto']  # Suponha que
'id_produto' seja o alvo do modelo

# Criar e treinar o modelo de regressão linear
modelo = LinearRegression()
modelo.fit(X_treino, y_treino)

# Supondo que você já tenha treinado o modelo com os dados
originais e agora quer fazer previsões para novos itens do
pedido
# Definir as características dos novos itens do pedido
X_novos_dados = itens_pedido_df[['quantidade',
'preço_unitário']]

# Fazer previsões para os novos dados
previsoes = modelo.predict(X_novos_dados)

# Exibir as previsões
print("Previsões para os novos itens do pedido:")
print(previsoes)
```

Avaliação da performance dos modelos preditivos:

```
# Avaliar desempenho
y_verdadeiro = itens_pedido_df['id_produto']  # Suponha que
'id_produto' seja o valor verdadeiro

# Calcular o erro médio quadrático
erro_medio_quadratico = mean_squared_error(y_verdadeiro,
previsoes)
print(f'Erro Médio Quadrático: {erro_medio_quadratico}')
```

Comparação entre diferentes abordagens de análise preditiva:

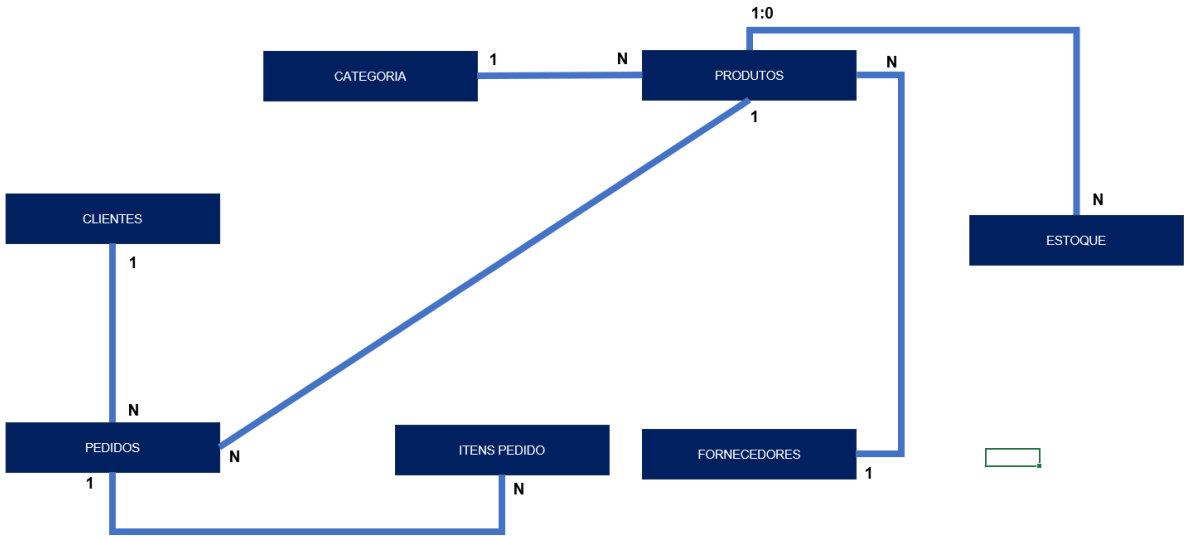
```
# Preparar os dados para os modelos
X1 = itens_pedido_df[['quantidade', 'preço_unitário']]
X2 = itens_pedido_df[['quantidade', 'id_produto']]
y = itens_pedido_df['id']

# Instanciar os modelos de regressão linear
modelo1 = LinearRegression()
modelo2 = LinearRegression()

# Ajustar os modelos aos dados
modelo1.fit(X1, y)
modelo2.fit(X2, y)

# Fazer previsões
previsoes1 = modelo1.predict(X1)
previsoes2 = modelo2.predict(X2)

# Avaliar os modelos
erro_medio_quadratico1 = mean_squared_error(y, previsoes1)
erro_medio_quadratico2 = mean_squared_error(y, previsoes2)

# Comparar modelos
if erro_medio_quadratico1 < erro_medio_quadratico2:
    print('Modelo 1 é melhor.')
else:
    print('Modelo 2 é melhor.')
```

3.3 Modelagem de dados

Para acessar o repositório do GitHub clique aqui

3.3.1 - Entrega 1: Modelagem Conceitual

Criação de um diagrama de entidade-relacionamento (ER) para representar os dados da empresa:



Identificação das entidades, atributos e relacionamentos relevantes:

Categorias

Clientes

Estoque

Fornecedores

Produtos

Pedidos

Itens_Pedido

3.3.2 - Entrega 2: Modelagem Lógica e Normalização

Transformação do modelo conceitual em um modelo lógico:

Categorias:

id (chave primária)

nome

Clientes:

id (chave primária)

nome

email

telefone

endereço

Estoque:

id (chave primária, FK para Produtos)

id_produto

quantidade

Fornecedores:

id (chave primária)

nome

contato

Produtos:

id (chave primária)

nome

descriçao

preço

id_categoria (chave estrangeira para Categorias)

Pedidos:

id (chave primária)

id_cliente (chave estrangeira para Clientes)

data_pedido

total

Itens_pedido:

id (chave primária)

id_pedido (chave estrangeira para Pedidos)

id_produto (chave estrangeira para Produtos)

quantidade

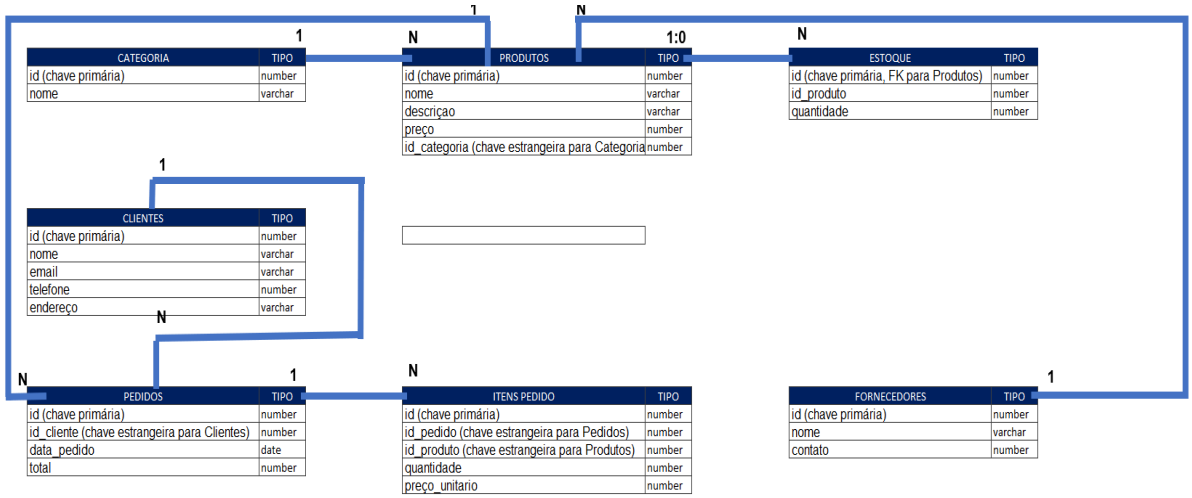
preço_unitario

Aplicação de técnicas de normalização para garantir a integridade dos dados:

Categorias, Clientes, Fornecedores e Produtos estão na Primeira Forma Normal

Estoque, Pedidos e Itens_Pedido estão na Segunda Forma Normal

Itens_pedido está na Terceira Forma Formal



3.3.3 - Entrega 3: Entregar Dicionário de Dados uma simulação de cadastro:

Mostrar as tabelas do Banco de Dados:

Tabela Categoria

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada categoria.	Primária
nome	varchar	Nome da categoria.	-

Tabela Clientes

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada cliente.	Primária
nome	varchar	Nome do cliente.	-
email	varchar	Endereço de email do cliente.	-
telefone	Int	Número de telefone do cliente.	-
endereço	varchar	Endereço completo do cliente.	-

Tabela Pedidos

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada pedido.	Primária
id_cliente	Int	Liga o pedido ao cliente que o realizou na tabela	Estrangeira
data_pedido	date	Data em que o pedido foi feito.	-
total	Int	Valor total do pedido	-

Tabela Produtos

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada produto.	Primária
nome	varchar	Nome do produto.	-
descriçao	varchar	Descrição detalhada do produto.	-
preço	Int	Preço unitário do produto.	-
id_categoria	Int	Liga o produto à sua categoria na tabela	Estrangeira

Tabela Itens_pedidos

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada item de pedido.	Primária
id_pedido	Int	Liga o item de pedido ao pedido correspondente	Estrangeira
id_produto	Int	Liga o item de pedido ao produto correspondente na tabela	Estrangeira
quantidade	Int	Quantidade do produto pedida neste item.	-
preço_unitario	Int	Preço unitário do produto neste item.	-

Tabela Estoque

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada registro de estoque.	Primária
id_produto	Int	Liga o registro de estoque ao produto correspondente na tabela	-
quantidade	Int	Quantidade do produto em estoque.	-

Tabela Fornecedores

Atributo	Tipo	Descrição	Chave
id	Int	Identificador único para cada fornecedor.	Primária
nome	varchar	Nome do fornecedor.	-
contato	Int	Informação de contato do fornecedor (telefone, email, etc.).	-

Fazer o registro nas tabelas do Banco de Dados (Simulação de Registro do Banco):

Tabela Categoria

id	nome
1	Categoria 1
2	Categoria 2
3	Categoria 3
4	Categoria 4
5	Categoria 5
6	Categoria 6
7	Categoria 7
8	Categoria 8
9	Categoria 9
10	Categoria 10

Tabela Clientes

id	nome	email	telefone	endereço
1	Cliente 1	cliente1@exemplo.com	(11) 99999-0001	Endereço do Cliente 1
2	Cliente 2	cliente2@exemplo.com	(11) 99999-0002	Endereço do Cliente 2
3	Cliente 3	cliente3@exemplo.com	(11) 99999-0003	Endereço do Cliente 3
4	Cliente 4	cliente4@exemplo.com	(11) 99999-0004	Endereço do Cliente 4
5	Cliente 5	cliente5@exemplo.com	(11) 99999-0005	Endereço do Cliente 5
6	Cliente 6	cliente6@exemplo.com	(11) 99999-0006	Endereço do Cliente 6
7	Cliente 7	cliente7@exemplo.com	(11) 99999-0007	Endereço do Cliente 7
8	Cliente 8	cliente8@exemplo.com	(11) 99999-0008	Endereço do Cliente 8
9	Cliente 9	cliente9@exemplo.com	(11) 99999-0009	Endereço do Cliente 9
10	Cliente 10	cliente10@exemplo.com	(11) 99999-00010	Endereço do Cliente 10

Tabela Estoque

id	id_produto	quantidade
1	1	20
2	2	63
3	3	48
4	4	63
5	5	8
6	6	9
7	7	31
8	8	54
9	9	23
10	10	43

Tabela Fornecedores

id	nome	contato
1	Fornecedor 1	contato1@exemplo.com
2	Fornecedor 2	contato2@exemplo.com
3	Fornecedor 3	contato3@exemplo.com
4	Fornecedor 4	contato4@exemplo.com
5	Fornecedor 5	contato5@exemplo.com
6	Fornecedor 6	contato6@exemplo.com
7	Fornecedor 7	contato7@exemplo.com
8	Fornecedor 8	contato8@exemplo.com
9	Fornecedor 9	contato9@exemplo.com
10	Fornecedor 10	contato10@exemplo.com

Tabela Itens_pedido

id	id_pedido	id_produto	quantidade	preço_unitário
1	686	3208	8	23,93
2	3268	3184	1	60,48
3	2751	1053	1	85,16
4	3188	2955	5	73,63
5	1621	909	2	42,9
6	1791	107	8	68,95
7	3448	2831	7	35,48
8	4658	3600	6	72,31
9	209	1664	9	39,91
10	2889	547	8	44,74

Tabela Pedidos

id	id_cliente	data_pedido	total
1	426	2024-05-04 20:39:23	313,44
2	356	2024-04-20 20:39:23	375,81
3	2624	2024-04-26 20:39:23	85,73
4	4519	2024-05-09 20:39:23	473,67
5	4356	2024-04-29 20:39:23	94,84
6	3496	2024-05-02 20:39:23	56,11
7	3861	2024-05-10 20:39:23	379,3
8	3492	2024-05-10 20:39:23	100,62
9	4048	2024-05-10 20:39:23	246,83
10	2019	2024-05-10 20:39:23	234,9

Tabela Produtos

id	nome	descrição	preço	id_categoria
1	Produto 1	Descrição do produto 1	19,91	6
2	Produto 2	Descrição do produto 2	56,45	9
3	Produto 3	Descrição do produto 3	92,24	2
4	Produto 4	Descrição do produto 4	22,23	4
5	Produto 5	Descrição do produto 5	28,01	11
6	Produto 6	Descrição do produto 6	56,55	6
7	Produto 7	Descrição do produto 7	64,14	8
8	Produto 8	Descrição do produto 8	83,33	10
9	Produto 9	Descrição do produto 9	86	8
10	Produto 10	Descrição do produto 10	25,91	12

3.4 Rede de computadores

Para acessar o repositório do GitHub clique aqui

3.4.1 - Entrega 1: Montar a planta baixa de Rede da Empresa

Definir os departamentos:

Administração, Recursos Humanos, Desenvolvimento, Operações, Marketing

Definir os equipamentos que serão utilizados em cada departamento:

Administração:

- Servidor para armazenamento seguro de dados
- Computadores para analistas e contadores
- Impressoras para documentos e relatórios
- Calculadoras financeiras para operações contábeis
- Software de contabilidade e gestão financeira

Recursos Humanos:

- Servidor para armazenamento seguro de dados
- Computadores para gerentes e especialistas de RH
- Impressoras para contratos e documentos
- Scanner para digitalização de documentos de RH
- Software de gestão de recursos humanos

Desenvolvimento:

- Servidores para hospedagem de aplicações e armazenamento de dados
- Computadores para desenvolvedores e técnicos de suporte
- Software de gestão de TI e segurança (ex.: ferramentas de monitoramento, antivírus, sistemas de backup)

Operações:

- Computadores para representantes de vendas e gerentes
- Smartphones e tablets para mobilidade e acesso a CRM
- Impressoras para contratos e materiais de marketing
- Software de gestão de relacionamento com clientes (CRM)

Marketing:

- Computadores
- Software de Design Gráfico
- Câmeras e Equipamentos de Vídeo

- Software de Análise de Dados
- Impressoras

3.4.2 - Entrega 2: Configuração de IP de todos os equipamentos

Definir a classe de Rede:

Classe de Rede: Classe C (192.168.0.0)

Definir o padrão de rede de cada departamento:

Padrão de Rede:

Administração: 192.168.0.1 - 192.168.0.50

Recursos Humanos: 192.168.0.51 - 192.168.0.100

Desenvolvimento: 192.168.0.101 - 192.168.0.150

Operações: 192.168.0.102 - 192.168.0.151

Marketing: 192.168.0.103 - 192.168.0.152

3.5 Segurança da Informação

Para acessar o repositório do GitHub clique aqui

3.5.1 - Entrega 1: Análise de Riscos

Identificação e avaliação dos riscos de segurança para a empresa:

Com base nos departamentos listados segue a lista de 20 possíveis ameaças e vulnerabilidades:

1.Phishing

Vulnerabilidade: Falta de treinamento em segurança para reconhecer emails fraudulentos.

2. Malware

Vulnerabilidade: Uso de software desatualizado ou sem antivírus adequado.

3. Acesso Não Autorizado

Vulnerabilidade: Controle inadequado de acesso e senhas fracas.

4. Ransomware

Vulnerabilidade: Falta de backups regulares e sistemas desatualizados.

5. Erro Humano

Vulnerabilidade: Ausência de políticas e procedimentos claros e treinamento insuficiente.

6. Sabotagem

Vulnerabilidade: Falta de monitoramento de funcionários e controle de acesso físico.

7. Violação de Dados

Vulnerabilidade: Proteção insuficiente de dados sensíveis e falta de criptografia.

8. Falhas de Backup

Vulnerabilidade: Processos de backup inadequados e testes de recuperação inexistentes.

9. Vulnerabilidades de Software

Vulnerabilidade: Falta de atualização regular e patches de segurança.

10. DDoS (Negação de Serviço)

Vulnerabilidade: Infraestrutura de rede sem defesas adequadas contra ataques de volume.

11. Exploração de Zero-Day

Vulnerabilidade: Dependência de software sem patches ou atualizações para vulnerabilidades recém-descobertas.

12. Man-in-the-Middle (MitM)

Vulnerabilidade: Comunicações não criptografadas e redes inseguras.

13. Falhas de Criptografia

Vulnerabilidade: Implementação inadequada ou falta de criptografia em dados sensíveis.

14. Roubo de Equipamentos

Vulnerabilidade: Falta de medidas de segurança física e monitoramento.

15. Configurações Incorretas

Vulnerabilidade: Falta de revisão e auditoria regular das configurações de sistemas e redes.

16. Insider Threats

Vulnerabilidade: Monitoramento inadequado e falta de controle sobre as atividades dos funcionários.

17. Exploração de Redes Sociais

Vulnerabilidade: Falta de políticas de uso de redes sociais e controle sobre contas oficiais.

18. Violação de Direitos Autorais

Vulnerabilidade: Uso não autorizado de materiais protegidos por direitos autorais sem verificação.

19. Roubo de Propriedade Intelectual

Vulnerabilidade: Proteção insuficiente de documentos e projetos sensíveis.

20. Interrupção de Fornecimento

Vulnerabilidade: Dependência excessiva de fornecedores únicos sem planos de contingência.

Análise de vulnerabilidades e ameaças potenciais:

1. Phishing

Impacto: Médio

Probabilidade: Alta

2. Malware

Impacto: Alto

Probabilidade: Alta

3. Acesso Não Autorizado

Impacto: Alto

Probabilidade: Média

4. Ransomware

Impacto: Alto

Probabilidade: Média

5. Erro Humano

Impacto: Médio

Probabilidade: Alta

6. Sabotagem

Impacto: Alto

Probabilidade: Baixa

7. Violação de Dados

Impacto: Alto

Probabilidade: Média

8. Falhas de Backup

Impacto: Alto

Probabilidade: Média

9. Vulnerabilidades de Software

Impacto: Médio

Probabilidade: Alta

10. DDoS (Negação de Serviço)

Impacto: Médio

Probabilidade: Média

11. Exploração de Zero-Day

Impacto: Alto

Probabilidade: Baixa

12. Man-in-the-Middle (MitM)

Impacto: Alto

Probabilidade: Baixa

13. Falhas de Criptografia

Impacto: Alto

Probabilidade: Média

14. Roubo de Equipamentos

Impacto: Médio

Probabilidade: Média

15. Configurações Incorretas

Impacto: Médio

Probabilidade: Alta

16. Insider Threats

Impacto: Alto

Probabilidade: Baixa

17. Exploração de Redes Sociais

Impacto: Médio

Probabilidade: Média

18. Violação de Direitos Autorais

Impacto: Médio

Probabilidade: Baixa

19. Roubo de Propriedade Intelectual

Impacto: Alto

Probabilidade: Baixa

20. Interrupção de Fornecimento

Impacto: Alto

Probabilidade: Baixa

3.5.2 - Entrega 2: Implementação de Medidas de Segurança

Implementação de políticas de controle de acesso aos sistemas e dados:

Política de Acesso a Dados Financeiros:

Acesso aos dados financeiros restrito a membros do departamento financeiro. Somente funcionários autorizados com funções específicas têm permissão para visualizar ou modificar esses dados.

Política de Acesso Baseado em Funções (RBAC):

Permissões de acesso são concedidas com base na função ou cargo do funcionário. Apenas as funções necessárias para o desempenho do trabalho têm acesso aos dados e sistemas relevantes.

Política de Senhas Fortes:

Exige que todas as senhas tenham no mínimo 12 caracteres, incluindo letras maiúsculas, minúsculas, números e símbolos. Senhas devem ser alteradas a cada 90 dias.

Política de Autenticação Multifator (MFA):

Implementação de autenticação multifator para todas as contas de usuário, especialmente para acesso a sistemas críticos e dados sensíveis.

Política de Revisão de Acesso:

Revisão regular dos direitos de acesso dos funcionários. Acessos desnecessários ou não utilizados são removidos periodicamente.

Política de Acesso Remoto:

Acesso remoto aos sistemas da empresa só é permitido através de VPN segura e autenticada. Acesso remoto deve ser monitorado e registrado.

Política de Controle de Acesso Físico:

Acesso físico a áreas sensíveis, como salas de servidores e arquivos confidenciais, é restrito a pessoal autorizado e monitorado por sistemas de segurança.

Política de Acesso a Dados de RH:

Dados de recursos humanos, como informações pessoais dos funcionários, são acessíveis apenas por membros do departamento de RH e outros funcionários autorizados.

Política de Controle de Acesso Temporário:

Acesso temporário concedido a contratados e visitantes deve ser limitado no tempo e escopo, com supervisão direta e expiração automática.

Política de Registro e Monitoramento de Acessos:

Todos os acessos a sistemas críticos e dados sensíveis devem ser registrados e monitorados. Logs de acesso devem ser revisados regularmente para detectar atividades suspeitas.

Configuração de sistemas de detecção de intrusão e prevenção de ataques:

Sistema de Detecção de Intrusão (IDS):

Configuração de Alertas: Configurar alertas em caso de atividades suspeitas na rede, como tentativas de acesso não autorizado ou varreduras de porta.

Sistema de Prevenção de Intrusão (IPS):

Bloqueio Automático: Implementar um IPS que bloqueie automaticamente tráfego malicioso identificado, prevenindo ataques antes que eles possam causar danos.

Análise de Tráfego em Tempo Real: Monitorar o tráfego de rede em tempo real para identificar padrões de comportamento anômalo que possam indicar um ataque.

Regras de Firewall: Configurar regras de firewall para bloquear tráfego não autorizado e permitir apenas comunicações essenciais e seguras.

Divisão de Rede: Dividir a rede em segmentos menores para limitar a propagação de ataques e isolar sistemas críticos de acesso geral.

Permissões Restritas: Implementar controle de acesso baseado em funções para garantir que usuários tenham apenas as permissões necessárias para suas funções.

Revisão Regular: Analisar logs de sistemas e redes regularmente para identificar e responder rapidamente a atividades suspeitas.

Verificação Adicional: Implementar autenticação multifator para adicionar uma camada extra de segurança no acesso a sistemas críticos.

Manutenção Regular: Garantir que todos os sistemas e softwares estejam atualizados com os patches de segurança mais recentes para proteger contra vulnerabilidades conhecidas.

Testes Periódicos: Realizar testes de penetração e avaliações de vulnerabilidade regularmente para identificar e corrigir pontos fracos na segurança.