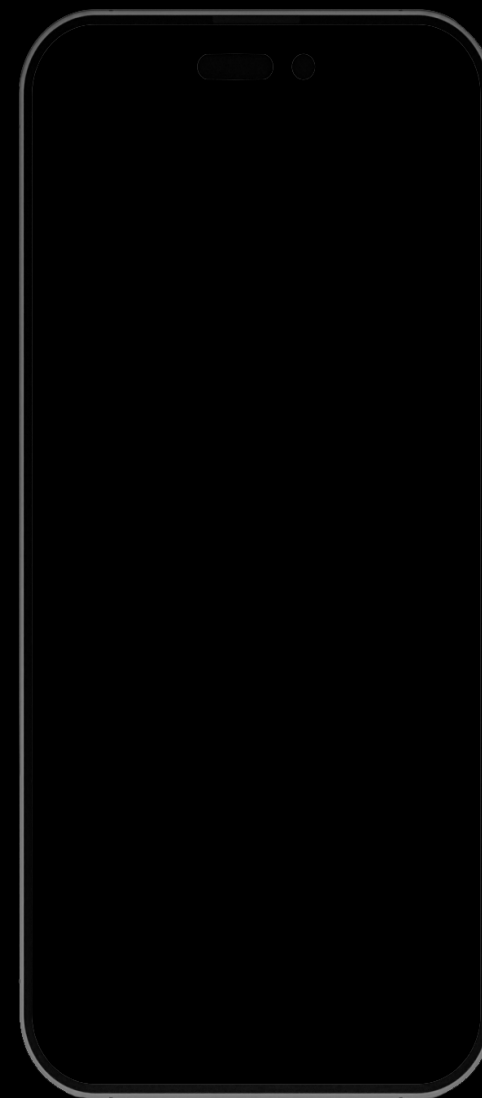


SANOFI

SANOFI
SYNC

TEMAS ESCOLHIDOS.





Tema 1

Avaliação de Desempenho

Fábrica da Medley em Campinas precisa de um sistema de avaliação de desempenho acessível aos operadores, atualmente realizadas em papel por falta de acesso a computadores.

Tema 9

Controle De Treinamentos

Time de Innovation & Development enfrenta dificuldades com planilha manual de controle de treinamentos, buscando solução para extração de dados e visualização clara de procedimentos críticos e pendências por colaborador e área



Tema 10

Controle de Movimentação de Documentos

Empresa busca otimizar a movimentação de documentação física entre setores para evitar extravios, economizar tempo e recursos, e envolver colaboradores em um processo mais confiável



PROPOSTA

SOLUÇÃO



Aplicativo Integrado para Funcionários e Gestores

Login: ao efetuar o login, dependendo da ID indicada, o usuário será levado para uma dessas variáveis da página:

- ✓ Seção dos outros Funcionários | Seção do Gestor;
- ✓ Ponto via QR code;
- ✓ Token para acessos (documentação da empresa);
- ✓ Treinamentos;
- ✓ Cadê Você?;
- ✓ Gestão de tempo (local com as horas do mês e organização das férias);
- ✓ Atestado;
- ✓ Feedback;



PROGRESSO

Foco em Código do Back-End



Melhorias Código Back-End

- ✓ Implementação de autenticação robusta baseada em JWT;
- ✓ Desenvolvimento de APIs RESTful;
- ✓ Integração com sistemas legados e APIs de terceiros.;
- ✓ Integração com sistemas legados e APIs de terceiros;
- ✓ Testes unitários para garantir a funcionalidade correta das APIs;

Novas Funcionalidades Adicionadas:

- ✓ Sistema de feedback contínuo;
- ✓ Geração de relatórios automáticos;

Melhorias Código Back-End

Frameworks/Bibliotecas Python:

- ✓ TensorFlow e Keras para desenvolvimento de modelos de Machine Learning;
- ✓ Scikit-Learn para análise de dados e algoritmos de aprendizado supervisionado e não supervisionado;

```
1 import tensorflow as tf
2 from tensorflow.keras.models import Sequential
3 from tensorflow.keras.layers import Dense
4 from sklearn.model_selection import train_test_split
5 from sklearn.datasets import load_iris
6 from transformers import pipeline
7 import ibm_watson
8 from ibm_watson import NaturalLanguageUnderstandingV1
9 from ibm_watson.natural_language_understanding_v1 import Features, SentimentOptions
10
11 # Dados
12 X, y = load_iris(return_X_y=True)
13 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
14
15 # Modelo Keras
16 model = Sequential([Dense(10, input_shape=(4,)), activation='relu'), Dense(3, activation='softmax')])
17 model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
18 model.fit(X_train, y_train, epochs=10, batch_size=1, verbose=0)
19 print(f'Acc: {model.evaluate(X_test, y_test, verbose=0)[1]:.2f}')
20
21 # ChatGPT
22 response = pipeline('text-generation', model='gpt2')("Importância do aprendizado de máquina.", max_length=50)
23 print(response[0]['generated_text'])
24
25 # IBM Watson
26 nlu = NaturalLanguageUnderstandingV1(version='2021-08-01', iam_apikey='YOUR_IBM_API_KEY', url='YOUR_IBM_URL')
27 analysis = nlu.analyze(text="Excelente desempenho, mas precisa melhorar a pontualidade.", features=Features(sentiment=SentimentOptions())).get_result()
28 print(analysis['sentiment'][0]['document']['label'])
29
30 # Google GEMINI (Simulado)
31 def google_gemini_integration(data):
32     return {"análise": "dados analisados com sucesso"}
33
34 print(google_gemini_integration(X_test))
35
```

Plataformas de IA:

- ✓ ChatGPT para implementação de assistente virtual
- ✓ IBM Watson para análise de sentimentos nos feedbacks;
- ✓ Google GEMINI para integração de dados e análises avançadas;
- ✓ TensorFlow/Keras: Utilizados para construir e treinar modelos de Machine Learning;
- ✓ Scikit-Learn: Implementado para análise de dados e previsões.

APIs Integradas

- ✓ Google Maps API para geolocalização e mapeamento;
- ✓ OpenWeather API para integração de dados meteorológicos.

Ferramentas de Desenvolvimento

- ✓ AWS para hospedagem e escalabilidade;
- ✓ Docker para criação de contêineres;
- ✓ Jenkins para integração e entrega contínua (CI/CD);
- ✓ GitHub para repositório de código e colaboração.

Análise de Desempenho: Utilização de modelos de Machine Learning para identificar padrões de desempenho.

Previsão de Desempenho: Desenvolvimento de modelos preditivos para antecipar desempenho futuro.

Análise de Sentimentos: Implementação de IA para análise de feedbacks, fornecendo insights sobre a moral e satisfação dos colaboradores.





VERSÃO | BETA



```
1 import numpy as np
2 from sklearn.model_selection import train_test_split
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5 from sklearn.datasets import load_iris
6 import tensorflow as tf
7 from tensorflow.keras.models import Sequential
8 from tensorflow.keras.layers import Dense
9 from transformers import pipeline
10 import ibm_watson
11 from ibm_watson import NaturalLanguageUnderstandingV1
12 from ibm_watson.natural_language_understanding_v1 import Features, SentimentOptions
13
14 # Análise de Desempenho: Treinamento do Modelo de Machine Learning
15 iris = load_iris()
16 X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_state=42)
17 model = Sequential([Dense(10, input_shape=(4,)), Dense(3, activation='softmax')])
18 model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
19 model.fit(X_train, y_train, epochs=10, batch_size=1, verbose=0)
20 accuracy = model.evaluate(X_test, y_test, verbose=0)[1]
21 print(f'Accuracy: {accuracy:.2f}')
22
23 # Previsão de Desempenho: Modelo Preditivo
24 X_train, X_test, y_train, y_test = train_test_split(np.arange(100).reshape(-1, 1), np.arange(100), test_size=0.2, random_state=42)
25 regressor = LinearRegression().fit(X_train, y_train)
26 predictions = regressor.predict(X_test)
27 mse = mean_squared_error(y_test, predictions)
28 print(f'Mean Squared Error: {mse:.2f}')
29
30 # Análise de Sentimentos: Utilização de IBM Watson
31 nlu = NaturalLanguageUnderstandingV1(version='2021-08-01', iam_apikey='YOUR_IBM_API_KEY', url='YOUR_IBM_URL')
32 analysis = nlu.analyze(text="Excelente desempenho, mas precisa melhorar a pontualidade.", features=Features(sentiment=SentimentOptions())).get_result()
33 print(analysis['sentiment']['document']['label'])
34
```

```
1 app = Flask(__name__)
2
3 # Gerar uma chave secreta aleatória
4 def generate_secret_key(length=24):
5     characters = string.ascii_letters + string.digits + string.punctuation
6     secret_key = ''.join(random.choice(characters) for i in range(length))
7     return secret_key
8
9 app.config['SECRET_KEY'] = generate_secret_key()
10
11 # Dados simulados
12 feedbacks = [
13     'Ótimo trabalho!',
14     'Precisa melhorar a pontualidade',
15     'Excelente desempenho',
16     'Bom, mas pode melhorar'
17 ]
18 performance_data = {
19     "Colaborador A": 80,
20     "Colaborador B": 85
21 }
22
23 # Middleware para verificar o token JWT
24 def token_required(f):
25     @wraps(f)
26     def decorated(*args, **kwargs):
27         token = request.args.get('token')
28         if not token:
29             return jsonify({'message': 'Token is missing!'}), 403
30         try:
31             jwt.decode(token, app.config['SECRET_KEY'], algorithms=["HS256"])
32         except:
33             return jsonify({'message': 'Token is invalid!'}), 403
34         return f(*args, **kwargs)
35     return decorated
36
37 # Rota de login para gerar token JWT
38 @app.route('/login', methods=['POST'])
39 def login():
40     auth = request.form
41     if auth and auth['username'] == 'user' and auth['password'] == 'password':
42         token = jwt.encode({'user': auth['username'], 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=30)}, app.config['SECRET_KEY'], algorithm="HS256")
43         return jsonify({'token': token})
44     return jsonify({'message': 'Could not verify!'}), 403
45
46 # Rota para obter dados de desempenho
47 @app.route('/performance', methods=['GET'])
48 @token_required
49 def get_performance():
50     return jsonify({'message': 'Desempenho coletado com sucesso', 'data': performance_data})
51
52 # Rota para obter feedbacks
53 @app.route('/feedbacks', methods=['GET'])
54 def get_feedbacks():
55     return jsonify(feedbacks)
56
57 # Rota para gerar relatório
58 @app.route('/report', methods=['GET'])
59 @token_required
60 def get_report():
61     report = {
62         "Total de Treinamentos": 50,
63         "Pendências": 5,
64         "Conclusões": 45
65     }
66     return jsonify(report)
67
68 if __name__ == '__main__':
69     app.run(debug=True)
70
```

json

```
[  
  "Ótimo trabalho!",  
  "Precisa melhorar a pontualidade",  
  "Excelente desempenho",  
  "Bom, mas pode melhorar"  
]
```

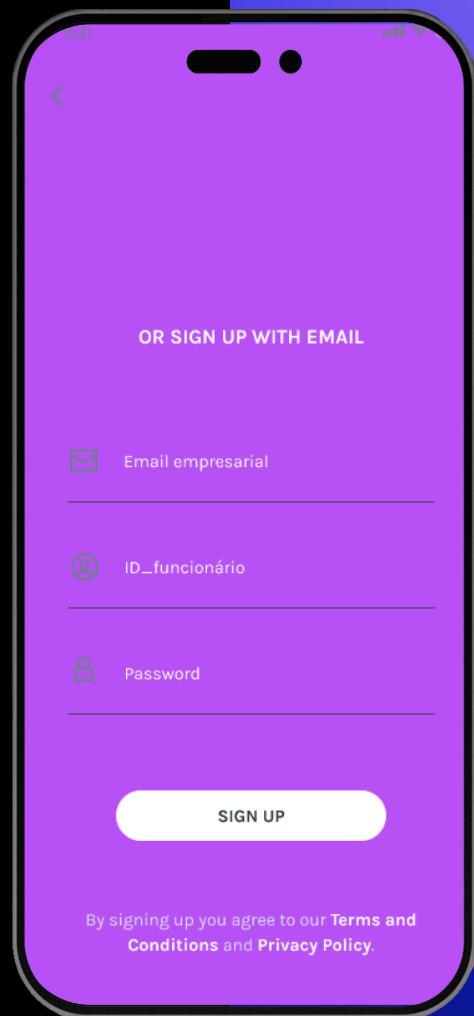
json

```
{  
  "message": "Desempenho coletado com sucesso",  
  "data": {  
    "Colaborador A": 80,  
    "Colaborador B": 85  
  }  
}
```

json

```
{  
  "Total de Treinamentos": 50,  
  "Pendências": 5,  
  "Conclusões": 45  
}
```

SINGUP



Smartphone mockup showing the Singup screen. The screen has a purple gradient background. At the top, there is a back arrow icon. Below it, the text "OR SIGN UP WITH EMAIL" is centered. There are three input fields: "Email empresarial" with an envelope icon, "ID_funcionário" with a person icon, and "Password" with a lock icon. A white "SIGN UP" button is at the bottom. At the very bottom, there is a small text line: "By signing up you agree to our Terms and Conditions and Privacy Policy."

OR SIGN UP WITH EMAIL

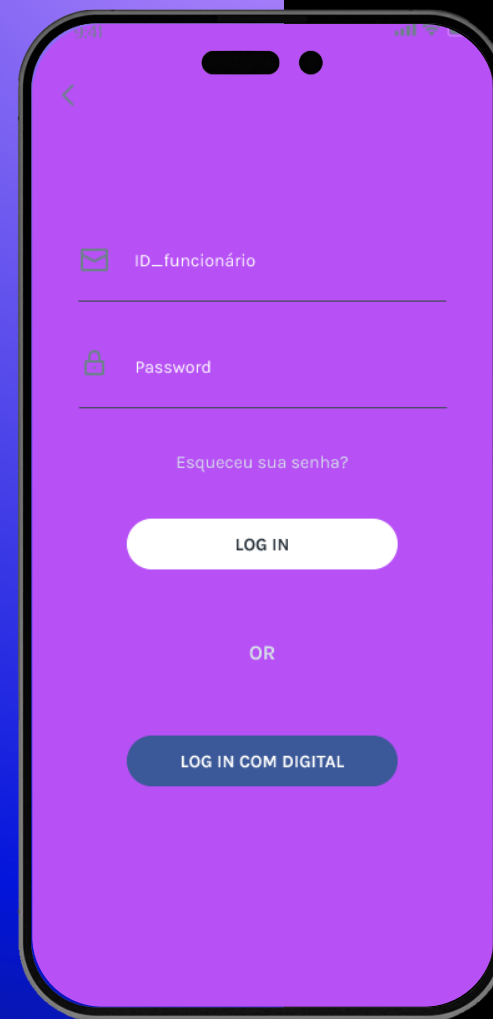
Email empresarial

ID_funcionário

Password

SIGN UP

By signing up you agree to our Terms and Conditions and Privacy Policy.



Smartphone mockup showing the Login screen. The screen has a purple gradient background. At the top, there is a back arrow icon. Below it, there are two input fields: "ID_funcionário" with an envelope icon and "Password" with a lock icon. Below these fields is a link "Esqueceu sua senha?". A white "LOG IN" button is centered below the link. Below that is the text "OR". At the bottom is a dark blue button with the text "LOG IN COM DIGITAL".

ID_funcionário

Password

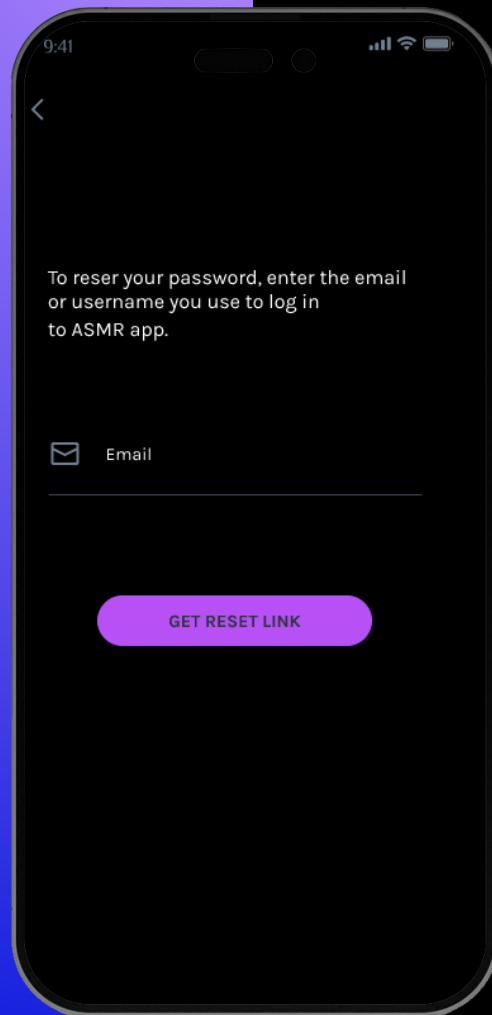
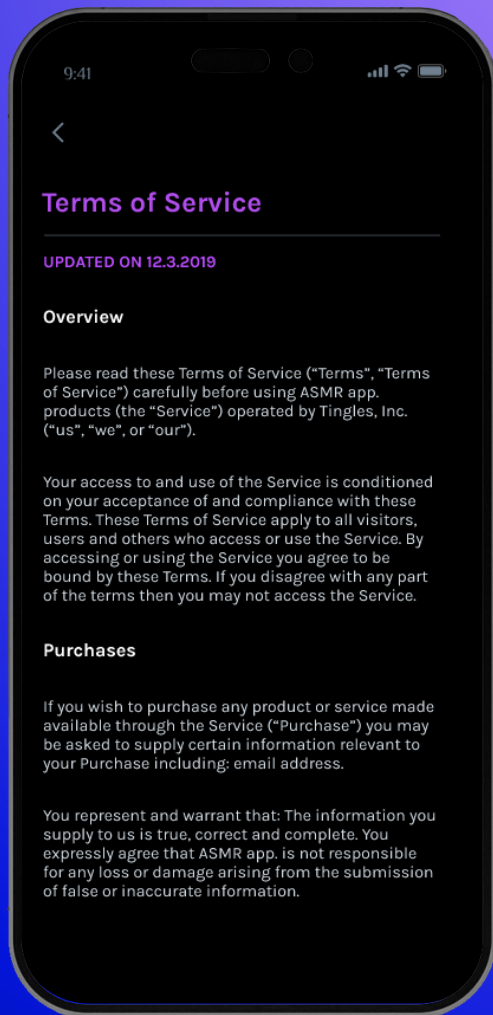
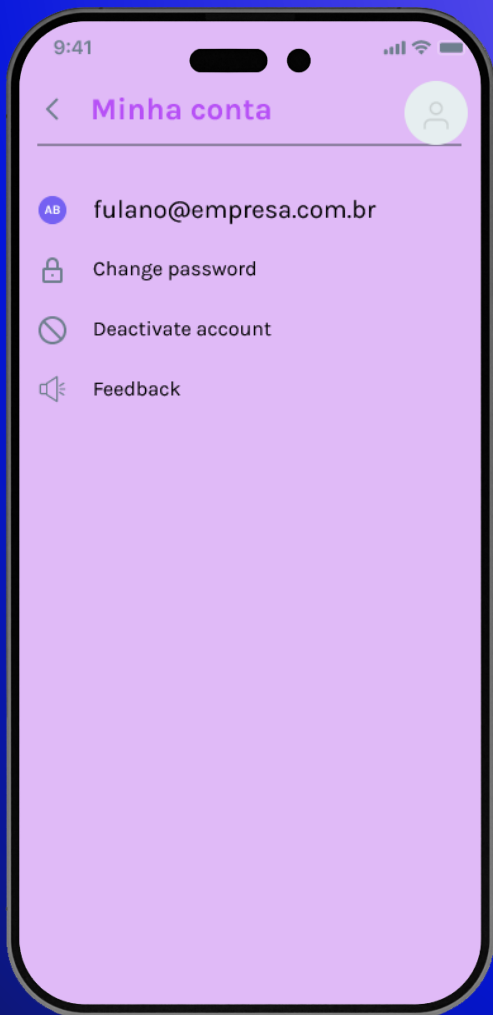
Esqueceu sua senha?

LOG IN

OR

LOG IN COM DIGITAL

LOGIN



01

RESET PASSWORD

02

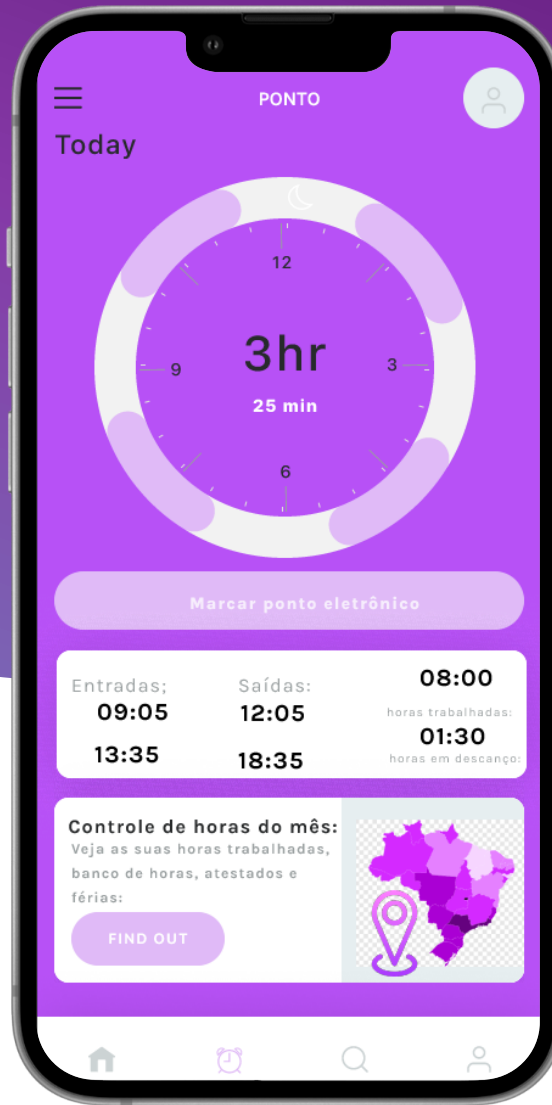
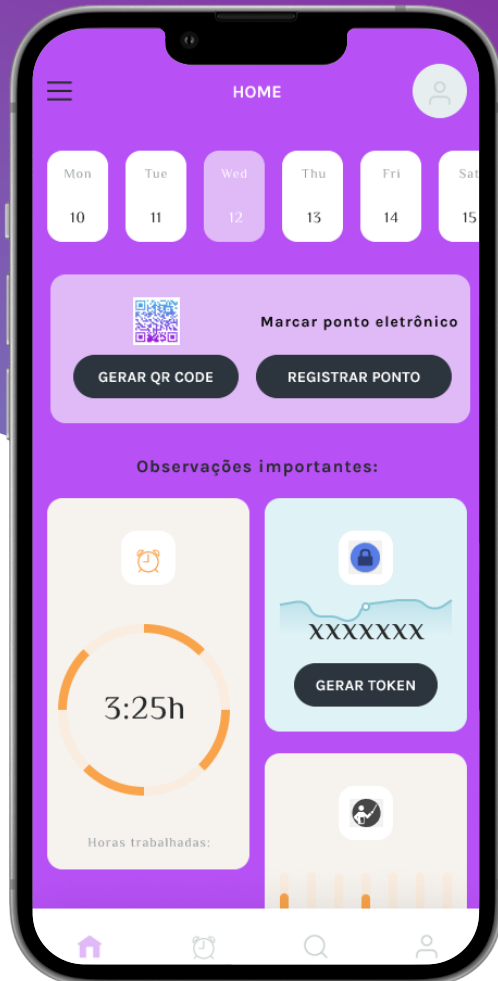
TERMS OF SERVICE

03

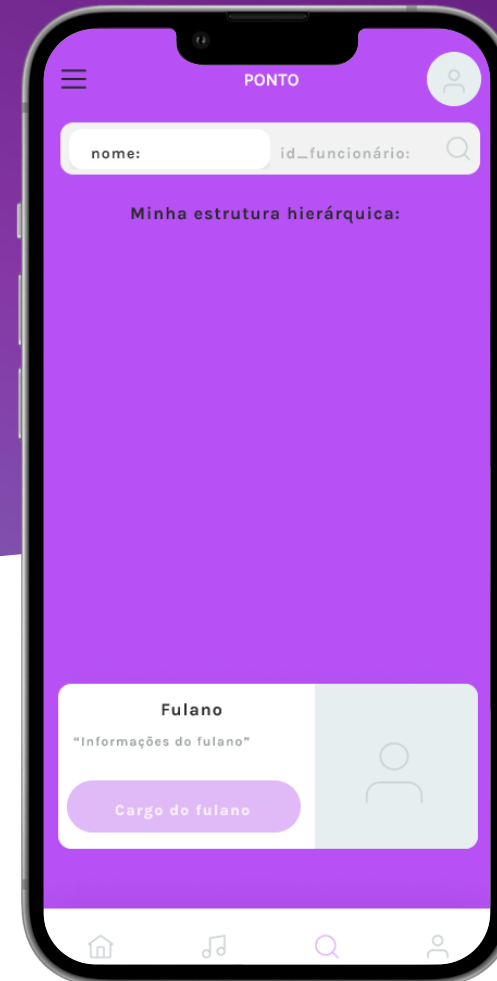
MY ACCOUNT

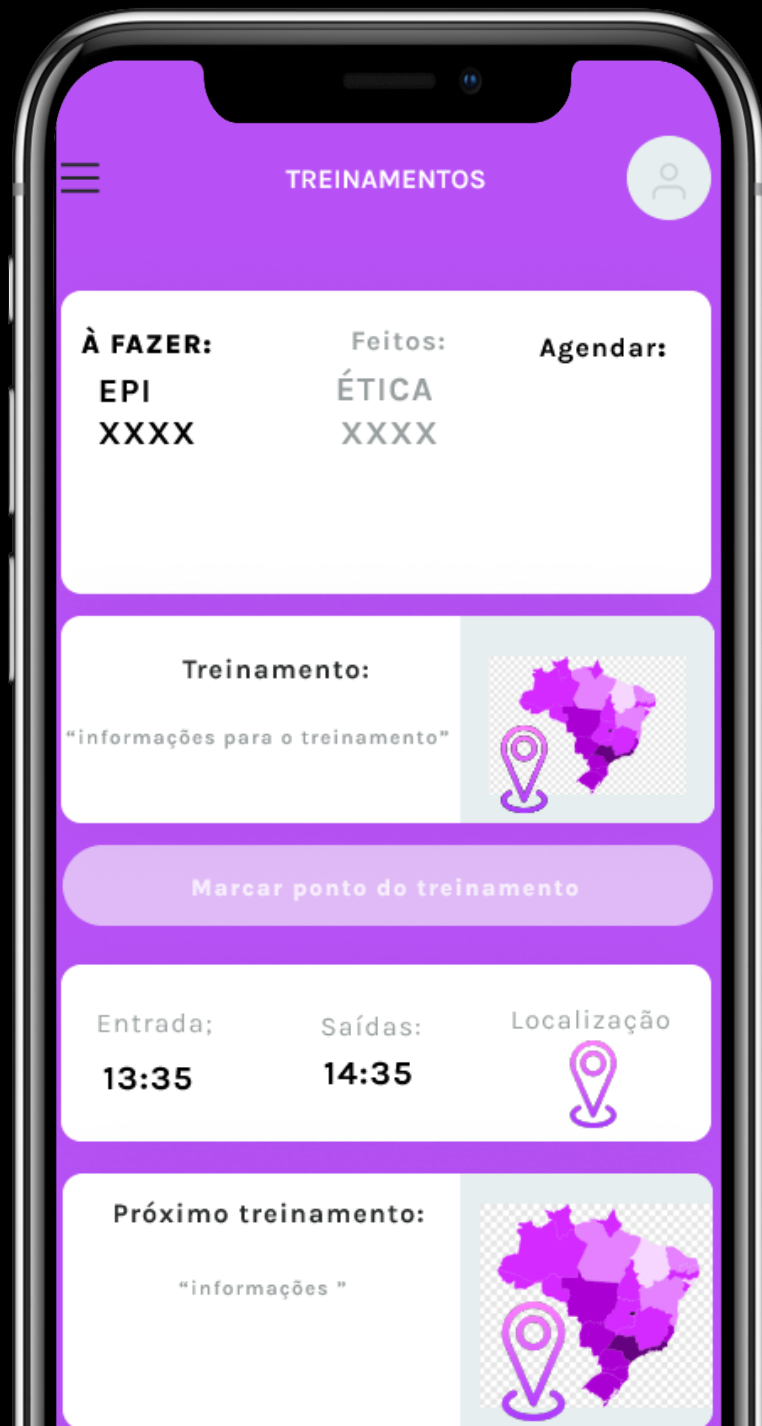
PONTO

HOME



PESQUISA





TREINAMENTOS

ABA EXCLUSIVA PARA GESTÃO



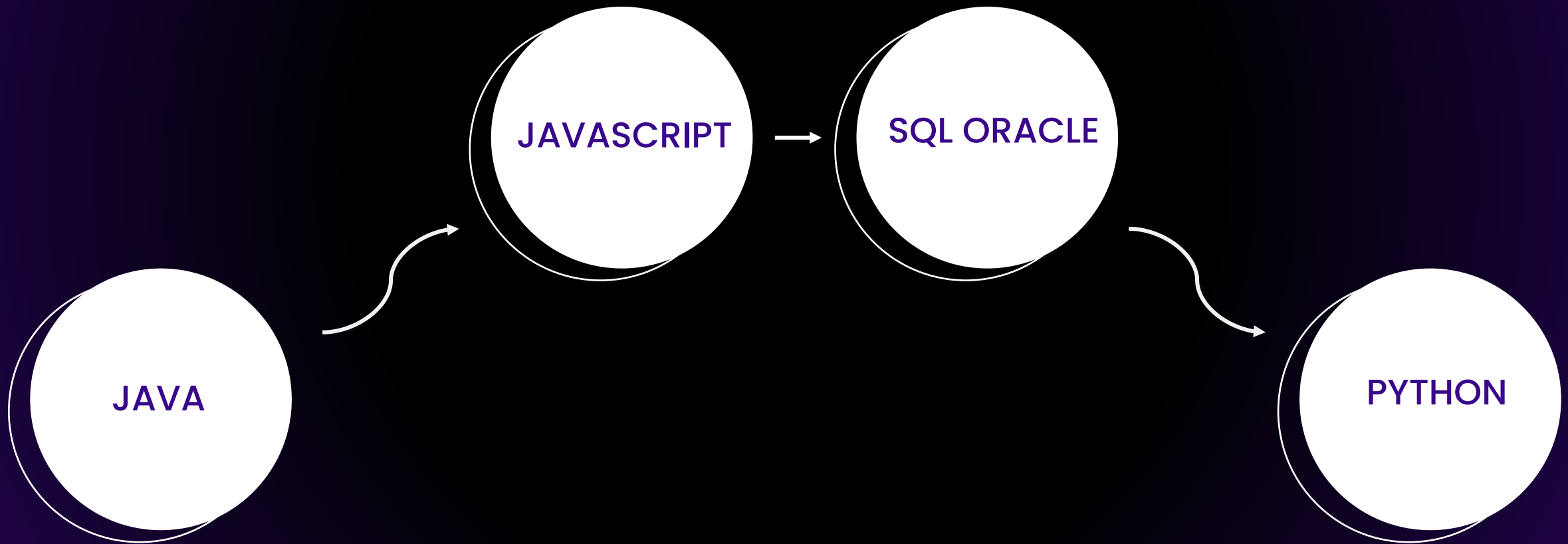


DESENVOLVIMENTO

Linguagens de Programação

LINGUAGENS DE PROGRAMAÇÃO

DESENVOLVIMENTO





CRONOGRAMA



PROVISÓRIO

2024



ABRIL

PLANNER

01	month Maio						
	S	M	T	W	T	F	S
			1	2	3	4	5
	6	7	8	9	10	11	12
	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28	29	30	31		

PLAN

Maio | Início do desenvolvimento dos bancos de dados com as informações necessárias e desenvolvimento de Back-End

Julho month							01
S	M	T	W	T	F	S	
					1	2	
3	4	5	6	7	8	9	
10	11	12	13	14	15	16	
17	18	19	20	21	22	23	
24	25	26	27	28	29	30	

PLAN

Julho | Desenvolvimento da parte de Front-End do aplicativo

01	month Junho						
	S	M	T	W	T	F	S
	1	2	3	4	5	6	7
	8	3	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31				

PLAN

Junho | Desenvolvimento da parte de Back-End do aplicativo e início do desenvolvimento Front-End;

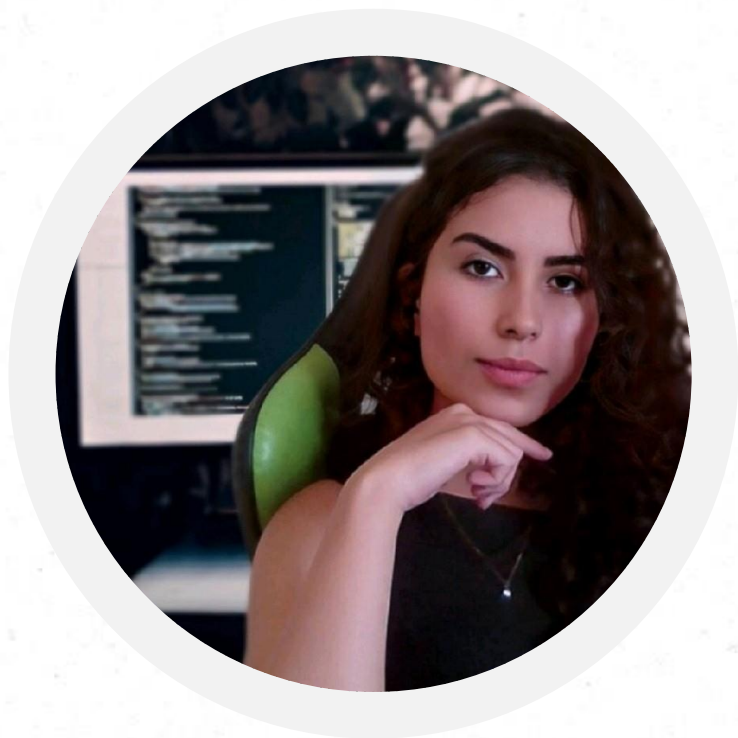
Agosto Setembro month							01
S	M	T	W	T	F	S	
			1	2	3	4	
5	6	7	8	9	10	11	
12	13	14	15	16	17	18	
19	20	21	22	23	24	25	
26	27	28	29	30	31		

PLAN

Agosto | Período de testagem do aplicativo e suas funções

Setembro | Correção de problemas e apresentação final do aplicativo

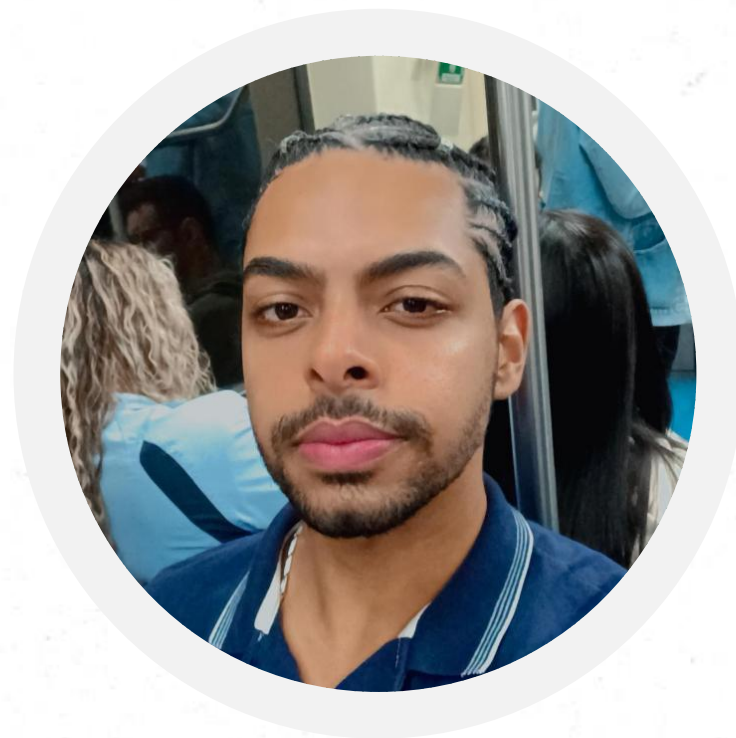
Nosso Time



Evelyn Cleto
Desenvolvedora
Full-Stack



Eduardo Kenji
Desenvolvedor
Full-Stack



Roberto Claudio
Desenvolvedor
Full-Stack

“ MUITO OBRIGADO ”