track.co

Componentes do Backend

FRAMEWORKS UTILIZADOS

-NestJS: framework principal para desenvolvimento backend em Node.js.

- Prisma: ORM para interagir com o banco de dados.
- Swagger: para documentação da API.

- Multer: para lidar com upload de arquivos.

TECNOLOGIAS UTILIZADAS

- TypeScript: linguagem de programação utilizada.

- Node.js: plataforma de execução.

- CSV-parser: biblioteca para processar arquivos CSV.

- NodeMailer: biblioteca para envio de emails (embora o código utilize nodemailer).

PADRÕES DE PROJETO IMPLEMENTADOS

Injeção de dependência

Utilizado pelo NestJS para gerenciar dependências entre classes

Repository

abstração para acesso ao banco de dados através do Prisma.

MVC

Arquitetura Model-View-Controller, implementada pelo NestJS.

Suite de Testes

TIPOS DE TESTES IMPLEMENTADOS

Testes de Carga

- Focados em testar unidades individuais de código (funções, classes) de forma isolada.
- Usamos o framework Jest para testes unitários no backend (NestJS)

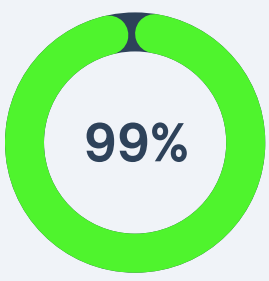
Testes de Integração

- Verificam a interação entre diferentes componentes do sistema.
- Usamos o framework Cypress para testes de integração, simulando interações do usuário com a interface e validando a comunicação com o backend.

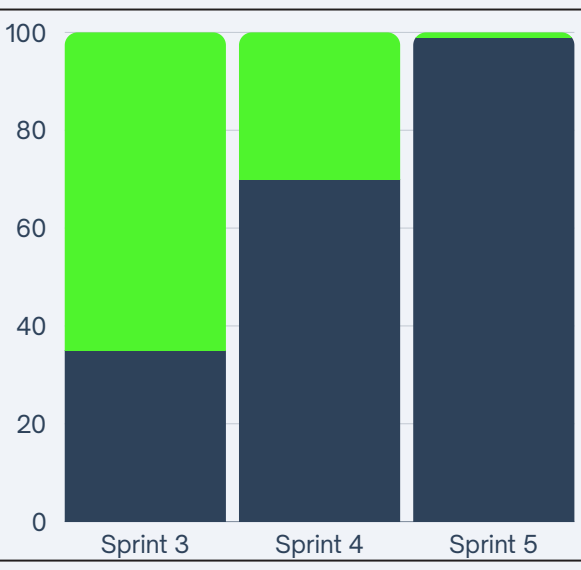
Testes de Unitarios

- Avaliaram o desempenho do sistema sob condições de carga simulada.
- Usamos o K6 para executar testes de carga em diferentes cenários, medindo métricas como tempo de resposta e taxa de erros.

PORCENTAGEM DE COBERTURA



Todos os endpoints foram testados, com todos os tipos de teste implementados



track.co

Componentes do FrontEnd

TECNOLOGIAS UTILIZADAS

React: biblioteca utilizada para criar o frontend

SASS: extensão do CSS utilizado para eficiência e organização

AUTH0: sistema de autenticação e login da plataforma

Cypress: ferramenta utilizada para testes

PADRÕES DE PROJETO IMPLEMENTADOS

Componentização

Elementos reutilizáveis para todo o projeto

Segregação de Páginas

Cada página possui suas próprias responsabilidades

Serviços

Reutilização de funcionalidades para toda a aplicação

DADOS SOBRE OS TESTES

SPRINT 3

TESTES DE INTEGRAÇÃO

- GESTÃO DE PESQUISAS E DISTRIBUIÇÃO DE PESQUISAS:
  - ENTRADA: ENVIO DE ARQUIVO CSV E INFORMAÇÕES ADICIONAIS PARA CRIAÇÃO DE UMA NOVA PESQUISA.
  - SAÍDA ESPERADA: SUCESSO NA CRIAÇÃO E ATUALIZAÇÃO DE PESQUISAS, COM RESPOSTA DO SISTEMA CONFIRMANDO A OPERAÇÃO.
  - RESULTADO: TODOS OS TESTES BÁSICOS DE GESTÃO DE PESQUISAS PASSARAM, EXCETO A CONSULTA DE TODAS AS PESQUISAS PELO FRONT-END DEVIDO A RESTRIÇÕES DE ACESSO.

TESTES DE CARGA

- NÃO APLICÁVEL PARA A SPRINT 3.

TESTES UNITÁRIOS

- TESTE DE IMPORTAÇÃO DE PLANILHA:
  - ENTRADA: UPLOAD DE PLANILHA PREENCHIDA COM DADOS DE CLIENTES SEGUINDO O MODELO.
  - SAÍDA ESPERADA: IMPORTAÇÃO BEM-SUCEDIDA PARA DADOS VÁLIDOS; MENSAGENS DE ERRO ESPECÍFICAS PARA DADOS INVÁLIDOS.
  - RESULTADO: TODOS OS CENÁRIOS PREVISTOS FORAM COBERTOS, ENFATIZANDO A IMPORTAÇÃO CORRETA E A VALIDAÇÃO DOS DADOS DOS CLIENTES.

ANÁLISE ESTÁTICA COM SONARQUBE

- NÃO APLICÁVEL PARA A SPRINT 3.

SPRINT 4

TESTES DE INTEGRAÇÃO

- MONITORAMENTO EM TEMPO REAL E REGISTRO DE RESULTADOS:
  - ENTRADA: FINALIZAÇÃO DA IMPORTAÇÃO DE DADOS DOS CLIENTES.
  - SAÍDA ESPERADA: O RESUMO EXIBE A CONTAGEM EXATA DE CLIENTES IMPORTADOS, INCLUINDO QUAISQUER ERROS OU AVISOS.
  - RESULTADO: AVALIAÇÃO BEM-SUCEDIDA DA CAPACIDADE DO SISTEMA DE MONITORAR EM TEMPO REAL E REGISTRAR OS RESULTADOS DA DISTRIBUIÇÃO DE PESQUISAS.

TESTES DE CARGA

- CARGA DE PICO:
  - ENTRADA: 1.000 USUÁRIOS SIMULTÂNEOS ENVIANDO RESPOSTAS ÀS PESQUISAS.
  - SAÍDA ESPERADA: SISTEMA MANTÉM FUNCIONALIDADE COM POSSÍVEIS DEGRADAÇÕES ACEITÁVEIS NO TEMPO DE RESPOSTA.
  - RESULTADO: TESTES REALIZADOS PARA ASSEGURAR QUE O SISTEMA OPERASSE EFICIENTEMENTE SOB CONDIÇÕES DE CARGA SIMULANDO O USO REAL.

TESTES UNITÁRIOS

- TESTE DE DISTRIBUIÇÃO DE PESQUISA:
  - ENTRADA: COMANDO PARA DISTRIBUIR UMA PESQUISA UTILIZANDO O CANAL DE E-MAIL.
  - SAÍDA ESPERADA: REGISTROS DA DISTRIBUIÇÃO SÃO GERADOS CONFORME ESPERADO, INCLUINDO QUANTIDADE DE ENVIOS E ENTREGAS.
  - RESULTADO: FOCO NA PRECISÃO DA IMPORTAÇÃO E DISTRIBUIÇÃO DAS PESQUISAS.

ANÁLISE ESTÁTICA COM SONARQUBE

- BACKEND:
  - PROBLEMA CRÍTICO: USO REDUNDANTE DE 'AWAIT' EM OPERAÇÃO NÃO-PROMISE.
  - CORREÇÃO ESPERADA: REFATORAÇÃO DO CÓDIGO PARA REMOVER O 'AWAIT' REDUNDANTE.
  - RESULTADO: ANÁLISE REVELOU 6 PROBLEMAS, COM 1 CRÍTICO, E RECOMENDOU MELHORIAS.

SPRINT 5

TESTES DE INTEGRAÇÃO

- NÃO ESPECIFICADO PARA A SPRINT 5.

TESTES DE CARGA

- NÃO ESPECIFICADO PARA A SPRINT 5.

TESTES UNITÁRIOS

-
- TESTE DE SALVAMENTO DOS RESULTADOS DA DISTRIBUIÇÃO:
  - ENTRADA: COMANDO PARA SALVAR OS RESULTADOS DA DISTRIBUIÇÃO DE PESQUISAS.
  - SAÍDA ESPERADA: OS RESULTADOS SÃO SALVOS DE FORMA PRECISA, PERMITINDO ANÁLISE FUTURA.
  - RESULTADO: CONTINUIDADE NA AVALIAÇÃO DA GESTÃO DE IMPORTAÇÃO E VALIDADE DE DADOS DO CLIENTE.

ANÁLISE ESTÁTICA COM SONARQUBE

-
- FRONTEND:
  - PROBLEMA DE SEVERIDADE MAIOR: CÓDIGO COMENTADO.
  - CORREÇÃO ESPERADA: REMOÇÃO DO CÓDIGO COMENTADO PARA LIMPEZA E MANUTENÇÃO DO CÓDIGO.
  - RESULTADO: ANÁLISE IDENTIFICOU 28 PROBLEMAS, COM SUGESTÕES ESPECÍFICAS PARA CORREÇÃO.

SEÇÃO SOBRE OS REQUISITOS

PRINCIPAIS REQUISITOS FUNCIONAIS

- **GESTÃO DE IMPORTAÇÃO E VALIDADE DE DADOS DO CLIENTE**

O sistema deve permitir download e envio de um modelo de planilha por e-mail, validar os dados dos clientes ao importá-los e, se houver erros, orientar sobre as correções necessárias antes da confirmação da importação.

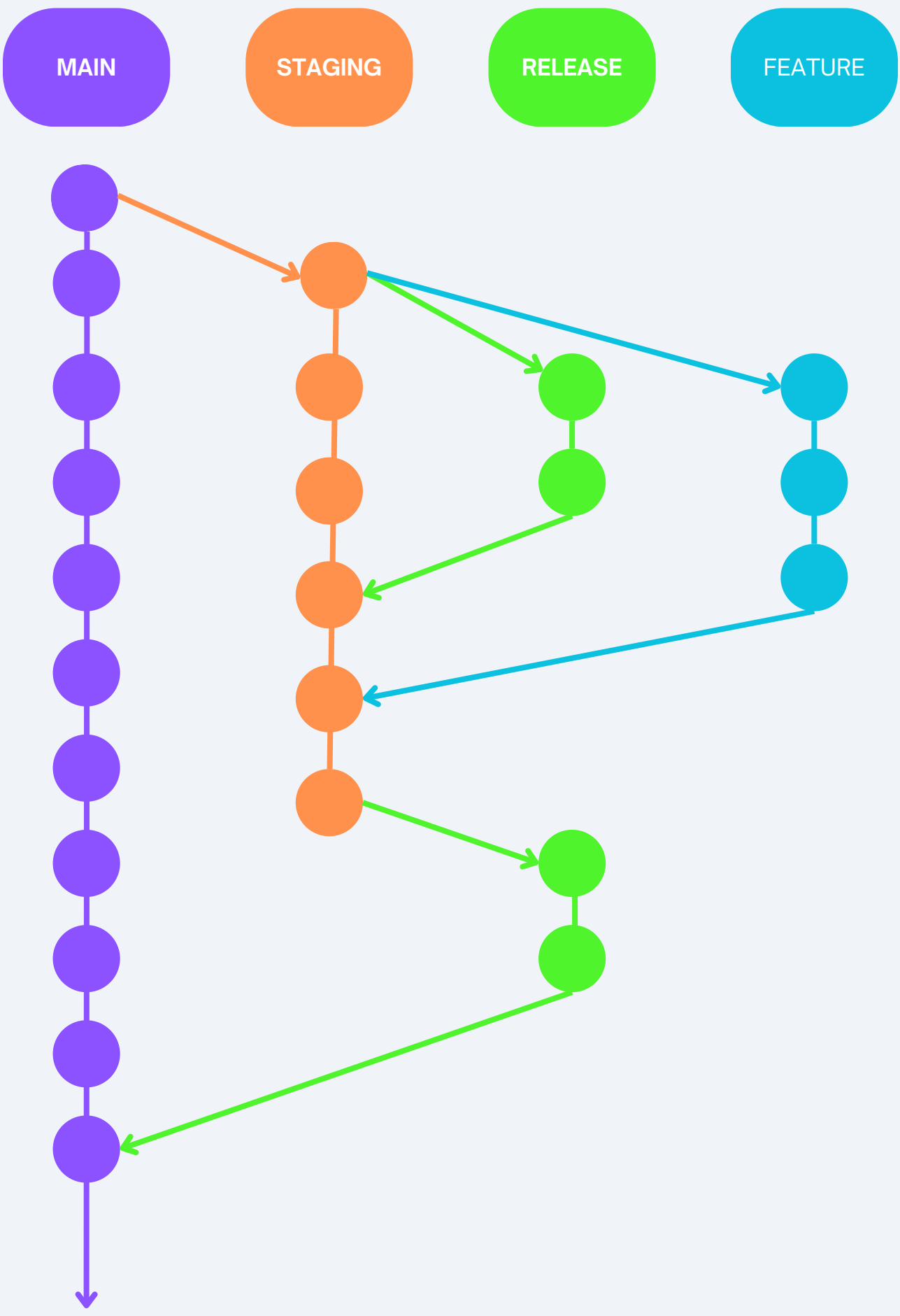
- **MONITORAMENTO EM TEMPO REAL E REGISTRO DE RESULTADOS DA DISTRIBUIÇÃO**

Após a importação, o sistema exibe um resumo com a quantidade de clientes para e-mails, permitindo verificação e salvamento dos resultados para futura análise.

- **VISUALIZAR O RESULTADO DA PESQUISA**

O sistema deve apresentar o resultado das pesquisas, exibindo informações agrupadas por resposta de usuários.

GIT FLOW



MAIN: Linha de vida principal do projeto, representando o código em produção. Atualizações nesta branch devem vir apenas da branch release

STAGING: A branch staging serve como um ambiente de preparação que imita a produção (main) o mais próximo possível. É útil para testes finais antes de um lançamento. A staging é atualizada com recursos da branch features ou diretamente da release para testes de pré-lançamento

RELEASE: A branch release é usada para preparar lançamentos. Ela é criada a partir da staging quando esta está estável e pronta para ser lançada. Ajustes finais, como correções de bugs e documentação de última hora, são feitos aqui.

FEATURE A estratégia para recursos permanece a mesma. Crie branches a partir da staging para novos recursos ou correções, usando a convenção /features/nome_do_recurso