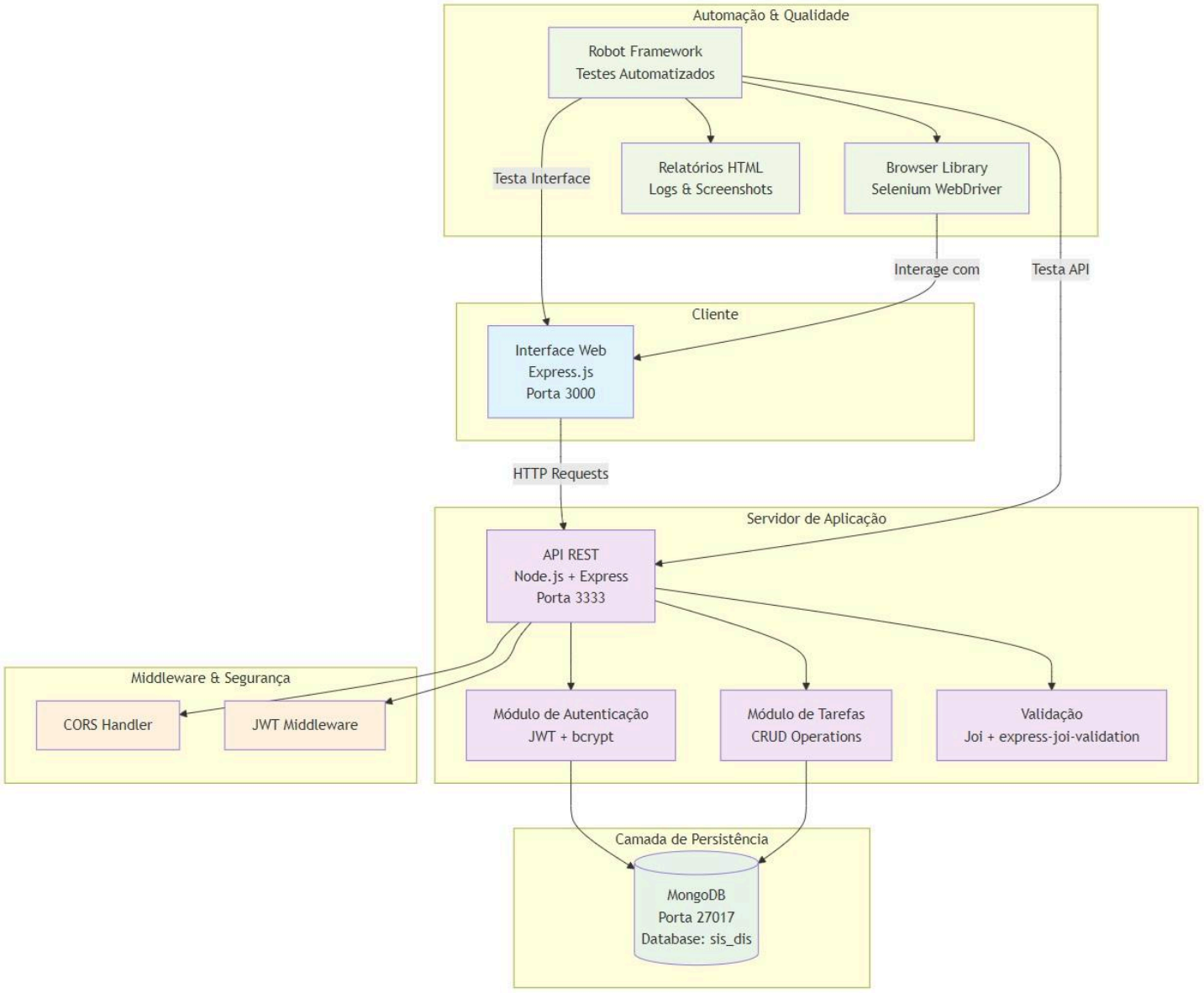
UNIVERSIDADE FEDERAL DO OESTE DO PARÁ
CAMPUS ORIXIMINÁ
Disciplina: Sistemas Distribuídos
Alunos: Marcelo Ferreira de Souza
Matrícula: 2021009010

Trabalho Final Sistemas Distribuidos (SisDis)

1. Projeto Final - Definição do Tema

O projeto SisDis é um sistema distribuído de gerenciamento de tarefas. O mesmo possui uma separação clara entre frontend e backend, utilizando comunicação via API REST e persistência distribuída com MongoDB. O mesmo implementa autenticação JWT para segurança e controle de acesso. A arquitetura distribuída garante escalabilidade horizontal, permitindo que diferentes componentes sejam executados em servidores distintos. O desenvolvimento será realizado utilizando Node.js para o backend (API REST), Express.js para o servidor web frontend, e MongoDB como banco de dados NoSQL distribuído.

2. Diagrama de Componentes



Descrição dos Componentes:

Frontend
Interface Web: Servidor Express.js que serve a aplicação frontend

Backend
API REST: Servidor Node.js que expõe endpoints para operações CRUD
Módulo de Autenticação: Gerencia login, registro e validação de tokens JWT
Módulo de Tarefas: Implementa a lógica de negócio para gerenciamento de tarefas
Validação: Middleware para validação de dados de entrada

Camada de Persistência
MongoDB: Banco de dados NoSQL para persistência distribuída

Middleware & Segurança
CORS Handler: Middleware para controle de acesso entre domínios
JWT Middleware: Middleware para autenticação e autorização

Automação & Qualidade
Robot Framework: Framework de testes automatizados end-to-end
Browser Library: Biblioteca para automação de navegador (Selenium WebDriver)
Relatórios HTML: Geração automática de relatórios, logs e screenshots

3. Projeto Final - Documento de Visão do Projeto (DVP)

3.1 Visão Geral do Projeto
O SisDis é um sistema distribuído de gerenciamento de tarefas projetado para demonstrar conceitos fundamentais de sistemas distribuídos em um ambiente prático e escalável.
3.2 Objetivos do Projeto
Implementar uma arquitetura de distribuída
Implementar persistência de dados em ambiente distribuído

3.3 Escopo do Projeto
3.3.1 Funcionalidades Incluídas
Sistema de autenticação de usuários
Gerenciamento completo de tarefas (CRUD)
Interface web responsiva
API REST documentada
Validação de dados
Controle de acesso baseado em tokens JWT

3.4 Quadro de Necessidades e Funcionalidades				
ID	Necessidade	Funcionalidade	Prioridade	Complexidade
N001	Autenticação de usuários	Sistema de login/registro com JWT	Alta	Média
N002	Gerenciamento de tarefas	CRUD completo de tarefas	Alta	Baixa
N003	Interface amigável	Frontend web responsivo	Alta	Média
N004	Segurança de dados	Validação e sanitização de	Alta	Baixa

		entrada		
N005	Persistência distribuída	Armazenamento em MongoDB	Alta	Baixa
N006	Comunicação entre serviços	API REST padronizada	Alta	Média
N007	Controle de acesso	Middleware de autorização	Média	Média
N008	Tratamento de erros	Sistema robusto de error handling	Média	Baixa
N009	Documentação da API	Endpoints documentados	Baixa	Baixa
N010	Logs do sistema	Sistema de logging distribuído	Baixa	Baixa

3.5 Arquitetura Técnica

3.5.1 Tecnologias Utilizadas

Backend: Node.js, Express.js, Mongoose
Frontend: HTML, CSS, JavaScript, Express.js (servidor estático)
Banco de Dados: MongoDB
Autenticação: JWT (JSON Web Tokens)
Validação: Joi
Segurança: bcrypt, CORS

3.5.2 Padrões Arquiteturais

Arquitetura de Microserviços: Separação clara entre frontend e backend
API REST: Comunicação padronizada via HTTP
MVC Pattern: Organização do código em camadas
Middleware Pattern: Processamento de requisições em pipeline

3.6 Requisitos Não Funcionais

3.6.1 Performance

Tempo de resposta da API < 500ms
Suporte a 100 usuários concorrentes
Disponibilidade de 99.5%

3.6.2 Segurança

Autenticação obrigatória para operações sensíveis
Validação de entrada em todos os endpoints
Tokens JWT com expiração configurável
Senhas criptografadas com bcrypt

3.6.3 Escalabilidade

Arquitetura preparada para balanceamento horizontal
Banco de dados distribuído
Separação de responsabilidades entre componentes

3.7 Cronograma de Desenvolvimento			
Fase	Atividade	Duração	Dependências
1	Configuração do ambiente	1 dia	-

2	Desenvolvimento da API	2 dias	Fase 1
3	Implementação da autenticação	1 dia	Fase 2
4	Desenvolvimento do frontend	3 dias	Fase 3

3.8 Riscos e Mitigações			
Risco	Probabilidade	Impacto	Mitigação
Falha na comunicação entre serviços	Média	Alto	Implementar retry logic e circuit breaker
Perda de dados	Baixa	Alto	Backup automático do MongoDB
Sobrecarga do servidor	Média	Médio	Implementar rate limiting
Vulnerabilidades de segurança	Baixa	Alto	Auditoria de segurança e testes

3.9 Critérios de Sucesso

- Sistema funcionando com todos os componentes integrados
- Autenticação e autorização implementadas corretamente
- Interface web responsiva e funcional
- API REST documentada e testada

3.10 Entregáveis

- Automação
- Backend
- Frontend
- Database