Estruturas de projeto de desenvolvimento

Exemplo prático com Loopback

Projeto de Desenvolvimento

- Estrutura de arquivos deve ser homogênea e clara
- Frameworks de desenvolvimento possuem estrutura similar
  - JavaStruts, RubyOnRails, Loopback, etc.
- Organização comum
  - Documentação : arquivos .txt/ .md / outros
  - Configurações: em diretório separado /config ou na raiz
    - “Config” , “/”
    - Arquivos Git
  - Aplicação servidora
    - “Server” , ”app”
  - Dependências comuns
    - “Common” , ”util”
  - Interface
    - “Client” , “pages”

Tarefa do arquiteto

- Entender a documentação
  - Nem sempre é tão completa quanto esperado
- Entender a estrutura de arquivos
  - Estrutura similar entre frameworks CRUD
- Fazer o mapeamento entre os conceitos e estrutura
  - Manual x arquivos
  - **Novamente**: o mapeamento pode não ser completo
  - Pode ser critério para não selecionar um framework
- Utilizando o framework Loopback 3.0
  - Comando **lb** é o gerador da aplicação

Arquivos no diretório raiz

- Dependências
  - Package.json (prática comum em servidores **node.js**): https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/package.json
  - /node_modules: local de instalação dos módulos
- Arquivos Git
  - .gitignore: https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/.gitignore
  - .gitmodules: https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/.gitmodules
- Documentação
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/README.md

Diretório server

- Configurações da API
  - component-config.json
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/component-config.json
  - Config.json https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/config.json
- Execução do servidor
  - Aplicação principal: **server.js**
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/server.js
  - Atualiza modelo de dados: **script.js**
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/script.js
- Scripts de inicialização
  - Autenticação
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/boot/authentication.js

Diretório server e common

- Conexão com fonte de dados
  - Comando **lb datasource**
  - Conectores
    - MongoDb, DB2, SQL Server, **PostgreSQL**, Redis, **SQLite3,** etc.
  - Datasource.development.json
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/datasources.development.json
    - Verificar propriedade **maxOfflineRequests**
- Conexão dos modelos criados com cada fonte
  - Comando **lb model**
  - Model-config.json
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/server/model-config.json
  - common/models/models.js
    https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/models.js

Criação do modelo de dados: common/models

- Arquivos JSON: definição dos modelos de dados

```
{
  "name": "myModel",
  "base": "PersistedModel",
  "properties": {
    // Properties listed here depend on your responses to the CLI
  },
  "validations": [],
  "relations": {},
  "acls": [],
  "methods": []
}
```

- Arquivos .js: scripts específicos por modelos
  - Rotas e/ou métodos incluídos manualmente, não suportados pelo framework
- Criados pelos comandos “*lb model*”, “*lb property*” ou “*lb relation*”

Exemplos concretos

- Bloco: apenas propriedades
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/bloco.json
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/bloco.js
- Departamento: propriedades e relações
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/departamento.json
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/departamento.js
- Disciplina: propriedades, relações e métodos
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/disciplina.json
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/disciplina.js
- Documentação em README.md específico
  - https://gitlab.c3sl.ufpr.br/ensalamento/ensalamento-back/blob/development/common/models/README.md

Criação de relações

- Mapeamento objeto relacional
  - Frameworks
    - Hibernate, Django, SQLAlchemy, ActiveRecord, Loopback, etc.
  - Mapeamentos “diretos”
    - Classes —> Tabelas
    - Atributos simples —> Colunas
  - Mapeamentos “complexos”
    - Relações: referências, composições, herança, etc.

BelongsTo

- Many-to-one, one-to-one
  - Definir chave estrangeira e chave primária (opcional)

```
Departamento —> Setor

relations": {
    "setor": {
      "type": "belongsTo",
      "model": "Setor",
      "foreignKey": "setorCod",
      // opcional “primaryKey” : “id”
    },
```

HasMany

- One-to-many: “Inversa” da BelongsTo
  - Definir chave estrangeira e chave primária (opcional)

```
Setor —> Departamento

"relations": {
    "departamentos": {
      "type": "hasMany",
      "model": "Departamento",
      "foreignKey": "setorCod"
    },
```

HasOne

- One-to-one: caso específico da *hasMany:* cardinalidade limitada
  - Define chave estrangeira e chave primária (opcional)
- Setor com um único Departamento (caso hipotético)

```
"relations": {
    "departamento": {
      "type": "hasOne",
      "model": "Departamento",
      "foreignKey": "setorCod"
    },
```

HasAndBelongsToMany

- Many-to-many
  - Cria tabela NxN (intermediária), sem acesso explícito
- Cursos e disciplinas
  - Em *Cursos*

```
"relations": {
    "disciplinas": {
      "type": "hasAndBelongsToMany",
      "model": "Disciplina"
    },
```

- Tabela Associativa *implícita :curso_disciplina (id_curso, id_disciplina)*

HasManyThrough

- Many-to-many
  - Cria tabela NxN (intermediária), sem acesso explícito
- Cursos e disciplinas + turma
  - Em *Cursos*

```
"relations": {
    "disciplinas": {
      "type": "hasMany",
      "model": "Disciplina",
      “through”: Turma
    }
```

- Em *Disciplina*

```
"relations": {
    "cursos": {
      "type": "hasMany",
      "model": "Cursos",
      “through”: Turma
    }
```

Agregações

- Semântica de agregação
  - EmbedsOne
  - EmbedsMany
  - EmbedsMany with belongsTo
  - Referencesmany
- EmbedsOne

```
{
  id: 1,
  name: 'John Smith',
  billingAddress: {
    street: '123 Main St',
    city: 'San Jose',
    state: 'CA',
    zipCode: '95124'
  }
}
```

EmbedsMany

```
{
  id: 1,
  name: 'John Smith',
  emailList: [{
    label: 'work',
    address: 'john@xyz.com'
  }, {
    label: 'home',
    address: 'john@gmail.com'
  }]
}
```

Relações polimórficas

- Relações para mais de um tipo de objeto
  - hasMany, belongsTo, hasOne, hasAndBelongsToMany, hasManyThrough
- Modificando a relação “professor” para “responsavel” (professor ou secretário)
- Turma

```
"responsavel": {
   "type": "belongsTo",
   "polymorphic": true
 },
```

Professor

```
"turmas": {
  "type": "hasMany",
  “model”: “Turma”,
  "polymorphic": {
       “selector”: “responsavel” }
},
```

OU

```
.. “polymorphic”: {
       “foreignKey”: “responsavelId”,
       “discriminator”: “responsavelType” }
},
```

Fazendo consultas

- As URLs podem ser usadas para criar consultas complexas
  https://loopback.io/doc/en/lb3/Querying-data.html
- Usando um JSON “stringfied”
  - /api/disciplinas?filter={%22where%22:{%22codigo%22:%22ta124%22}}
- Usando uma requisição REST-like
  - /api/disciplinas?filter[where][codigo]=ci1163

Sumário

- Frameworks possuem diversas funcionalidades
- Geração de CRUD
    - Servidor
    - Interface
- Outras
    - Autenticação
    - Validação
    - Mapas
    - Gestão de arquivos
    - Testes
    - Deployment