Aluno: Gusthor Sabino

Pesquisa detalhada sobre o desenvolvimento de sistemas no âmbito do back-end. O trabalho será estruturado conforme solicitado, com seções para introdução, desenvolvimento, considerações finais, referências bibliográficas e anexos com exemplos de códigos fonte.

Introdução

O desenvolvimento de sistemas no âmbito do back-end é uma área crucial na construção de aplicações modernas. Enquanto o front-end lida com a interface do usuário e a experiência visual, o back-end se concentra na lógica de negócios, armazenamento de dados e comunicação com outras partes do sistema. Neste trabalho, exploraremos as principais linguagens, tecnologias e arquiteturas utilizadas no desenvolvimento de back-end, bem como práticas recomendadas e exemplos de códigos fonte.

Desenvolvimento

1. Linguagens de Programação

1.1 JavaScript (Node.js)

- **História**: Introduzido em 2009, Node.js trouxe a execução do JavaScript para o lado do servidor.
- **Características**: Assíncrono, baseado em eventos, ideal para aplicações em tempo real.
- **Exemplo de Código**:

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(3000, '127.0.0.1', () => {
  console.log('Server running at http://127.0.0.1:3000/');
});
```

1.2 Python (Django, Flask)

- **História**: Python é uma linguagem versátil e fácil de aprender, com Django sendo lançado em 2005 e Flask em 2010.
- **Características**: Legibilidade, ampla biblioteca padrão, frameworks robustos.
- **Exemplo de Código (Flask)**:

```python
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello, World!'
if __name__ == '__main__':
    app.run()
```

1.3 Java (Spring Boot)

- **História**: Java, lançado pela Sun Microsystems em 1995, é amplamente usado em grandes sistemas corporativos. Spring Boot simplifica o desenvolvimento com Spring Framework.
- **Características**: Tipagem forte, segurança, desempenho.
- **Exemplo de Código (Spring Boot)**:

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
@RestController
class HelloController {
    @GetMapping("/")
    public String index() {
        return "Greetings from Spring Boot!";
    }
}
```

1.4 Ruby (Ruby on Rails)

- **História**: Lançado em 2005, Ruby on Rails popularizou o desenvolvimento rápido de aplicações web.
- **Características**: Convenção sobre configuração, DRY (Don't Repeat Yourself).
- **Exemplo de Código**:

```ruby
class ApplicationController < ActionController::Base
  def hello
    render html: "Hello, world!"
  end
end
```

2. Tecnologias

2.1 Bancos de Dados

- **SQL**: MySQL, PostgreSQL
    - **Exemplo**: `SELECT * FROM users WHERE id = 1;`
- **NoSQL**: MongoDB, Redis
    -

Exemplo (MongoDB):

```javascript
db.users.find({ "name": "John" });
```

2.2 Servidores Web

- **Apache**: Um dos servidores web mais populares.
- **Nginx**: Conhecido por sua alta performance e baixo consumo de recursos.

2.3 APIs

- **REST**: Arquitetura para construir APIs simples e escaláveis.
- **GraphQL**: Desenvolvido pelo Facebook, permite consultas mais flexíveis.
- **Exemplo (Express.js com REST):**

```javascript
const express = require('express');
const app = express();
app.get('/api/users', (req, res) => {
  res.send([{ name: 'John' }, { name: 'Jane' }]);
});
app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

3. Arquiteturas

3.1 Monolítica

- **Descrição: Sistema único onde todos os componentes são interligados.**
- **Vantagens: Simplicidade inicial.**
- **Desvantagens: Dificuldade de escalabilidade e manutenção.**

3.2 Microservices

- **Descrição: Arquitetura onde o sistema é dividido em serviços menores e independentes.**
- **Vantagens: Escalabilidade, independência de desenvolvimento e implantação.**
- **Desvantagens: Complexidade de gerenciamento, comunicação entre serviços.**
- **Exemplo: Cada microserviço pode ser uma API REST separada, conectando-se através de um gateway API.**

Considerações Finais

O desenvolvimento de sistemas no back-end é um campo vasto e em constante evolução. Com uma variedade de linguagens, tecnologias e arquiteturas, os desenvolvedores têm a flexibilidade de escolher as ferramentas que melhor se adequam às necessidades específicas de seus projetos. Embora cada escolha tenha suas vantagens e desvantagens, a combinação certa pode resultar em sistemas robustos, escaláveis e eficientes.

Referências Bibliográficas

1. **Node.js: nodejs.org**
2. **Flask: flask.palletsprojects.com**
3. **Spring Boot: spring.io/projects/spring-boot**
4. **Ruby on Rails: rubyonrails.org**
5. **MongoDB: mongodb.com**
6. **Express.js: expressjs.com**

Anexos

Exemplo de Código - Node.js

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(3000, '127.0.0.1', () => {
  console.log('Server running at http://127.0.0.1:3000/');
});
```

Exemplo de Código - Flask

```python
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello, World!'
if __name__ == '__main__':
    app.run()
```

Exemplo de Código - Spring Boot

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
@RestController
class HelloController {
    @GetMapping("/")
    public String index() {
        return "Greetings from Spring Boot!";
    }
}
```

Exemplo de Código - Ruby on Rails

```ruby
class ApplicationController < ActionController::Base
  def hello
    render html: "Hello, world!"
  end
end
```