FTCE

Faculdade de Tecnologia e Ciências Exatas

USJT - 2016 - ARQDESIS - Arquitetura e Desenvolvimento de Sistemas

Professores: Bonato, Élcio e Rodrigo

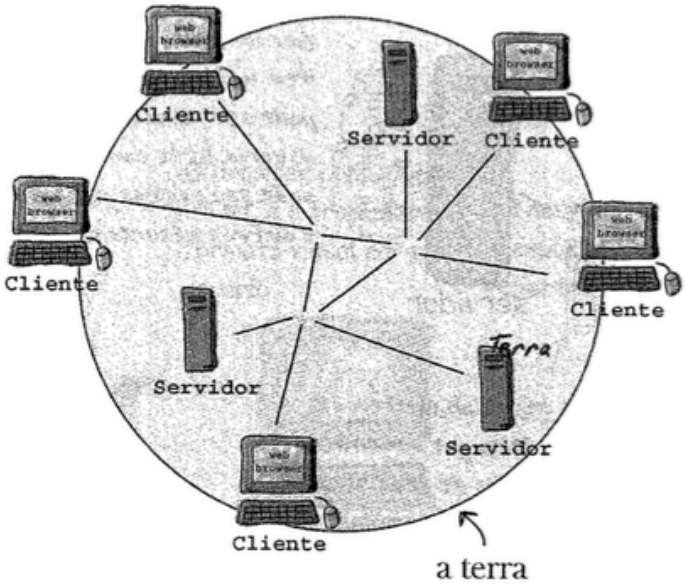
Aula: 01
Assunto: Protocolo HTTP, Design Patterns

Protocolo HTTP

Conceitos básicos

Arquitetura Web

A web consiste em zilhões de clientes (usando browsers como o Mozilla ou o Safari) e servidores (rodando aplicações como o Apache), conectados através de redes com fio e wireless. Nosso objetivo é construir uma aplicação que os clientes ao redor do mundo possam acessar. E nos tornarmos estupidamente ricos.



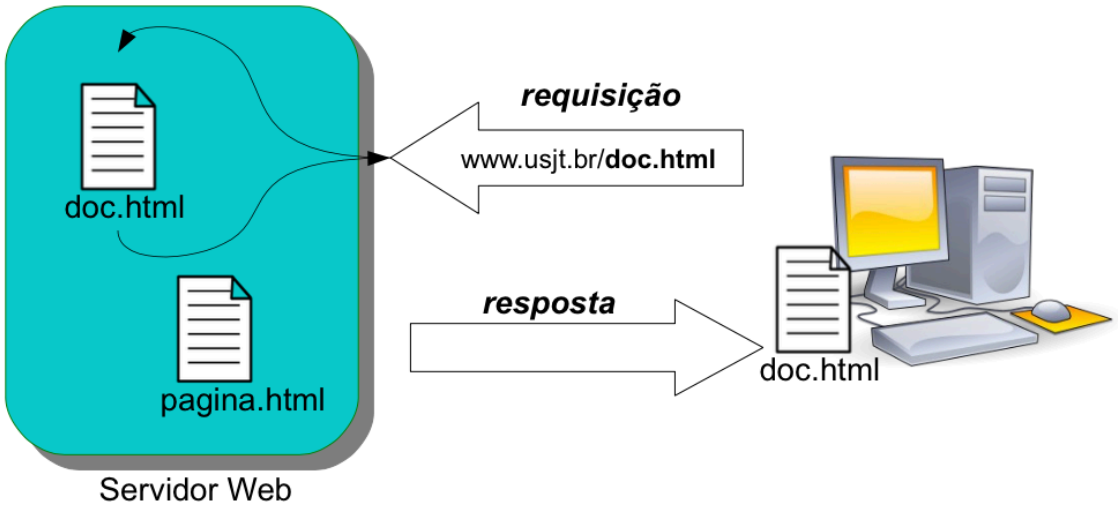
Fonte: (BASHAM, SIERRA, BATES 2008), p. 3

A arquitetura básica web é composta por 3 elementos:

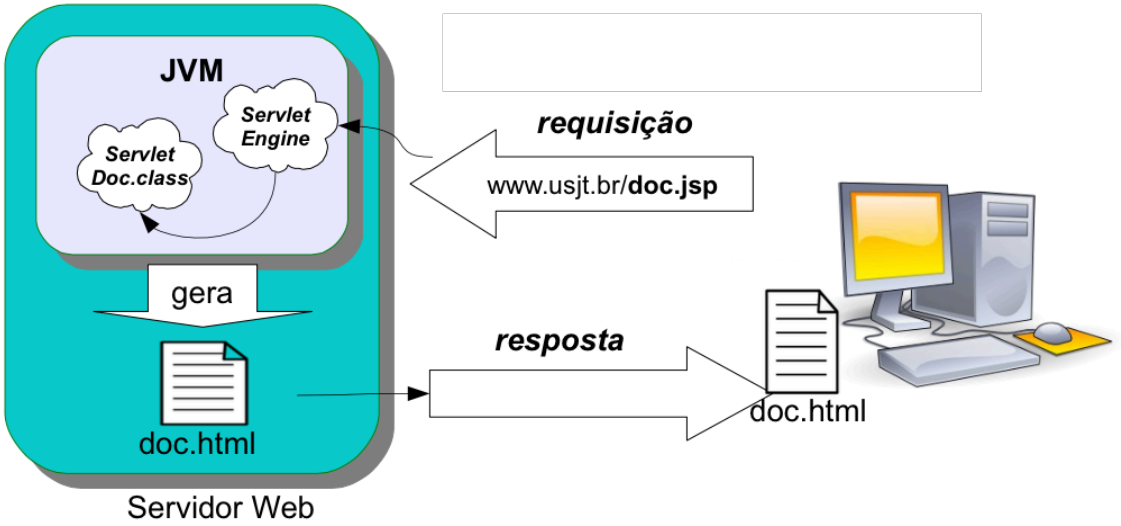
- Servidor Web
- Cliente (navegador web entende HTML)
- Protocolo de comunicação HTTP

Servidor Web

No passado, o conteúdo era apenas estático:



Com o passar do tempo surgiu a necessidade de se executar aplicações, isto é, de se ter conteúdo dinâmico.



Protocolo HTTP

Um **protocolo** é uma convenção ou padrão que controla conexão, comunicação ou transferência de dados; são regras que definem o formato das mensagens trocadas.

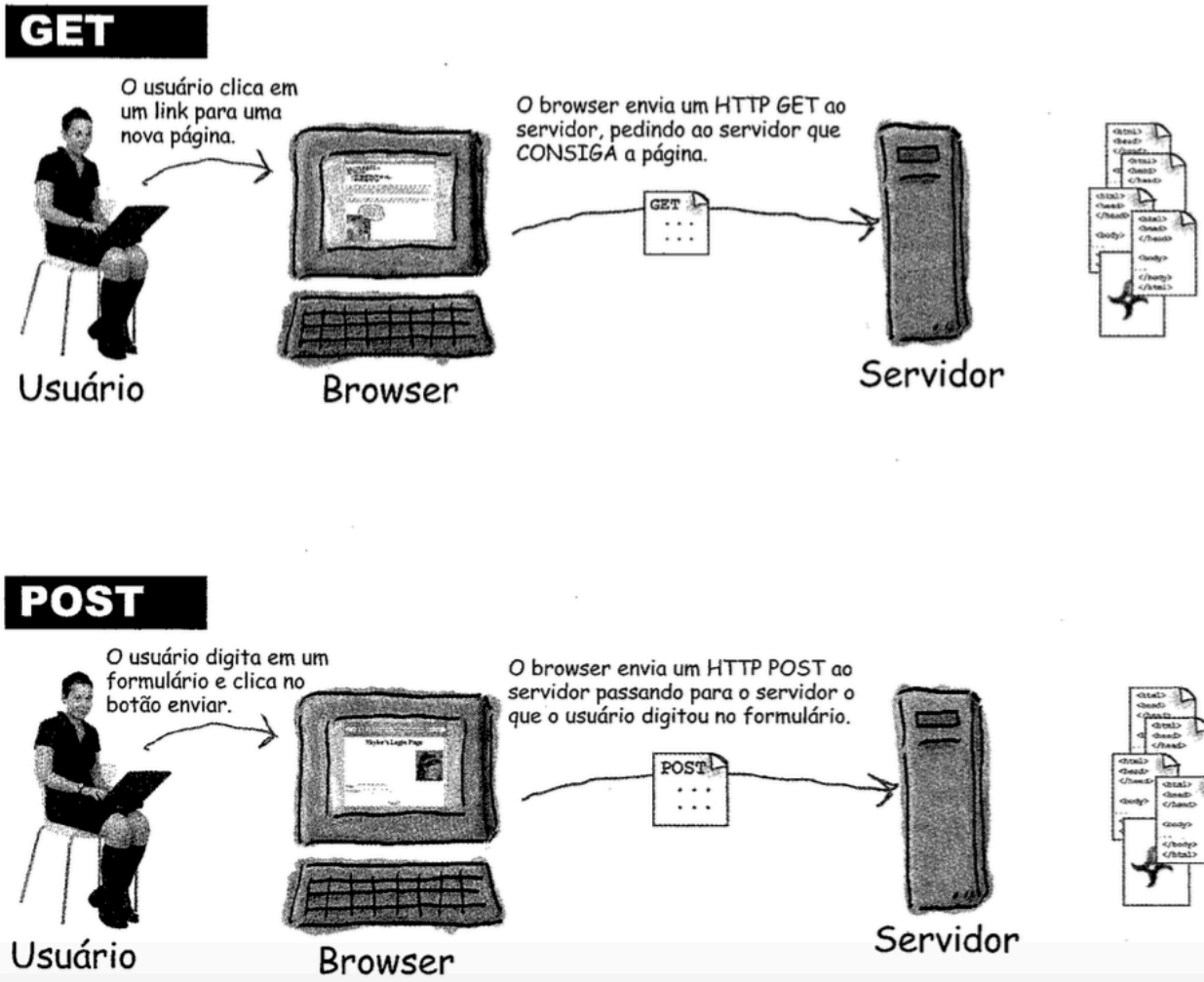
O **protocolo HTTP** é utilizado para comunicação entre cliente (navegador) e servidor Web. É baseado no modelo cliente/servidor ou **requisição/resposta** (request/response).

O HTTP é *stateless*, isto é, a princípio, nenhuma requisição é mantida no servidor.

Uma **requisição** consiste no envio de um pacote de dados HTTP solicitando ao servidor um determinado recurso (html, jsp, imagem, etc.). Deve conter um comando ou método que diz o que o servidor Web deverá fazer com a requisição.

Métodos mais importantes:

- GET → para obter um conteúdo do servidor
- POST → para enviar dados de formulário ao servidor



Fonte: (BASHAM, SIERRA, BATES 2008), p. 12

Diferenças

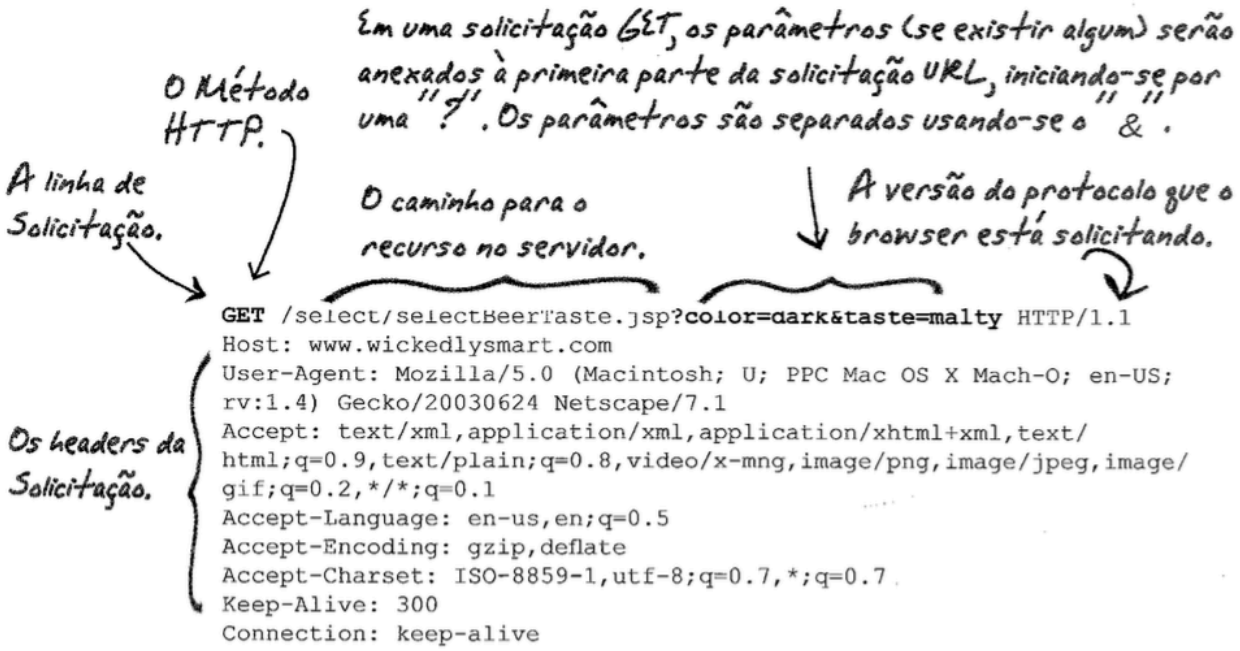
GET

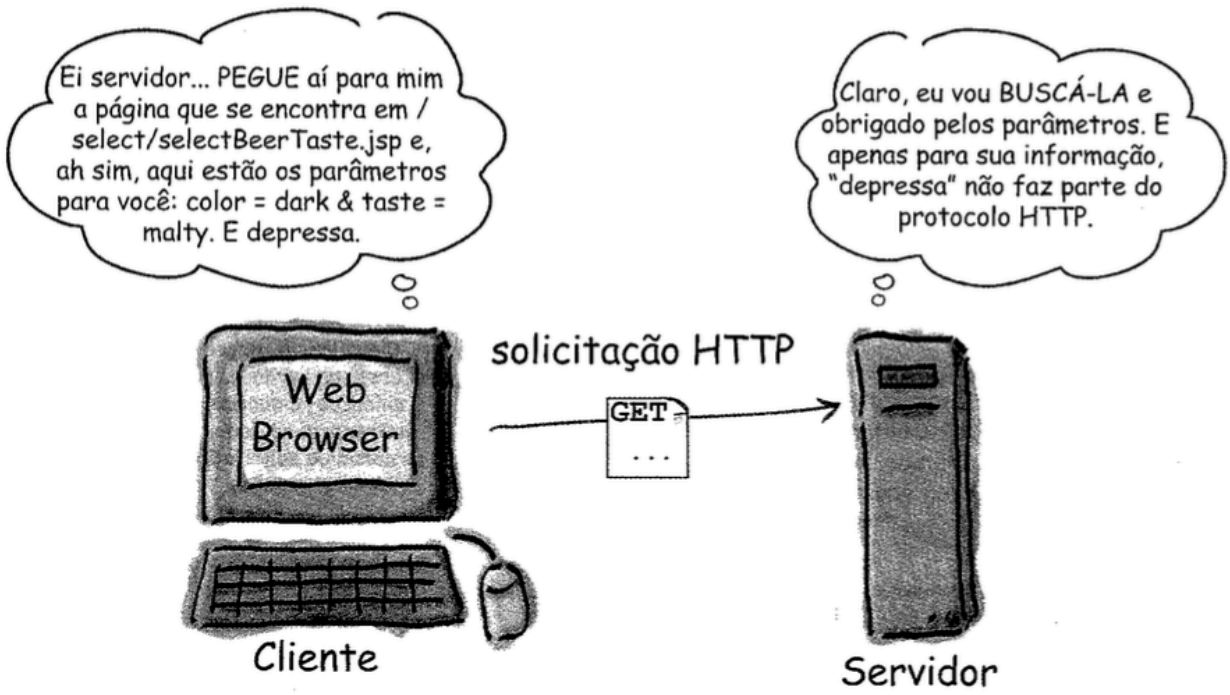
- Limitação de tamanho
- Dados são incluídos na URL (totalmente expostos)

POST

- Dados são incluídos no corpo da mensagem HTTP
- Sem limitação de tamanho

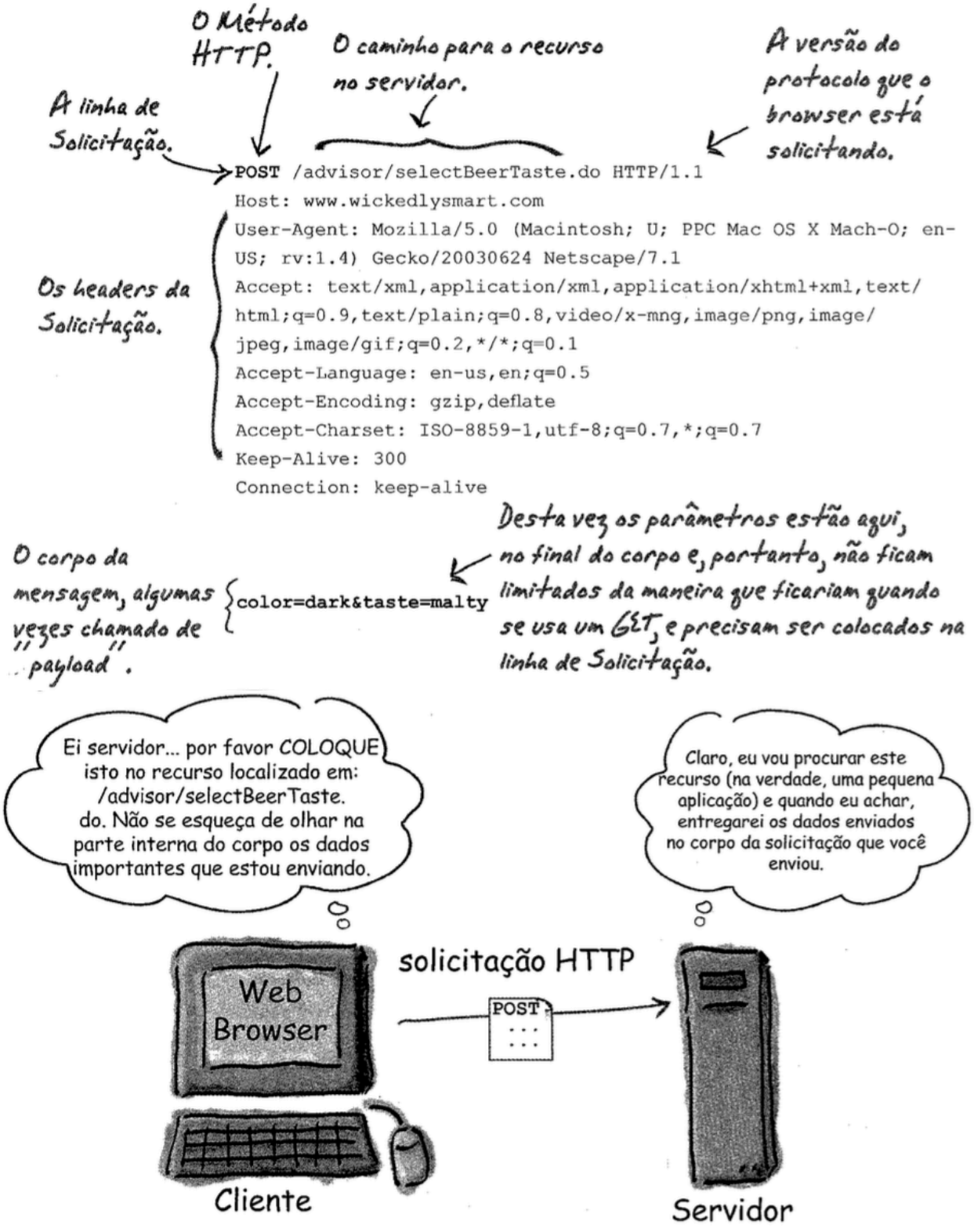
Formato da Requisição HTTP GET





Fonte: (BASHAM, SIERRA, BATES 2008), p. 15

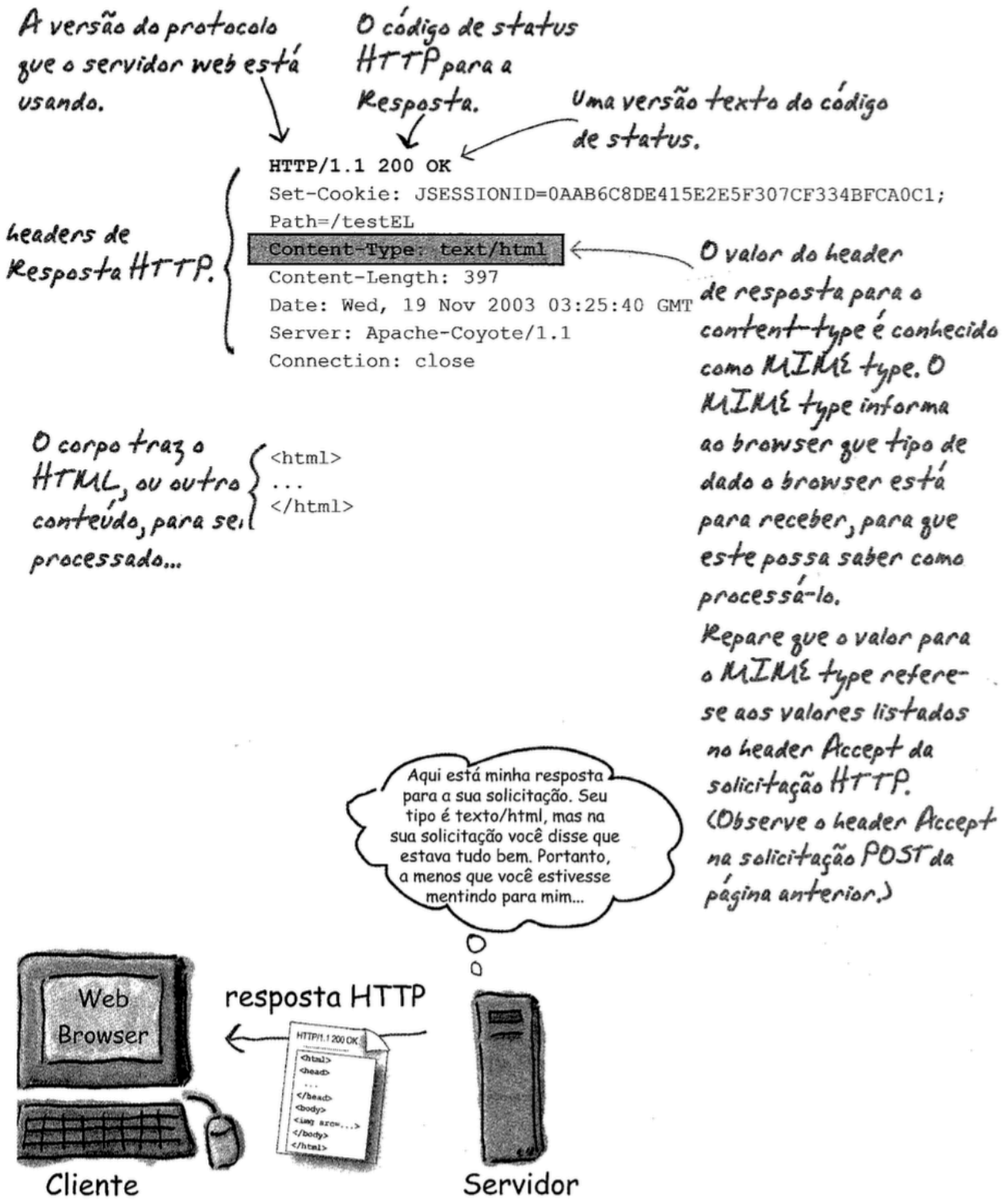
Formato da Requisição HTTP POST



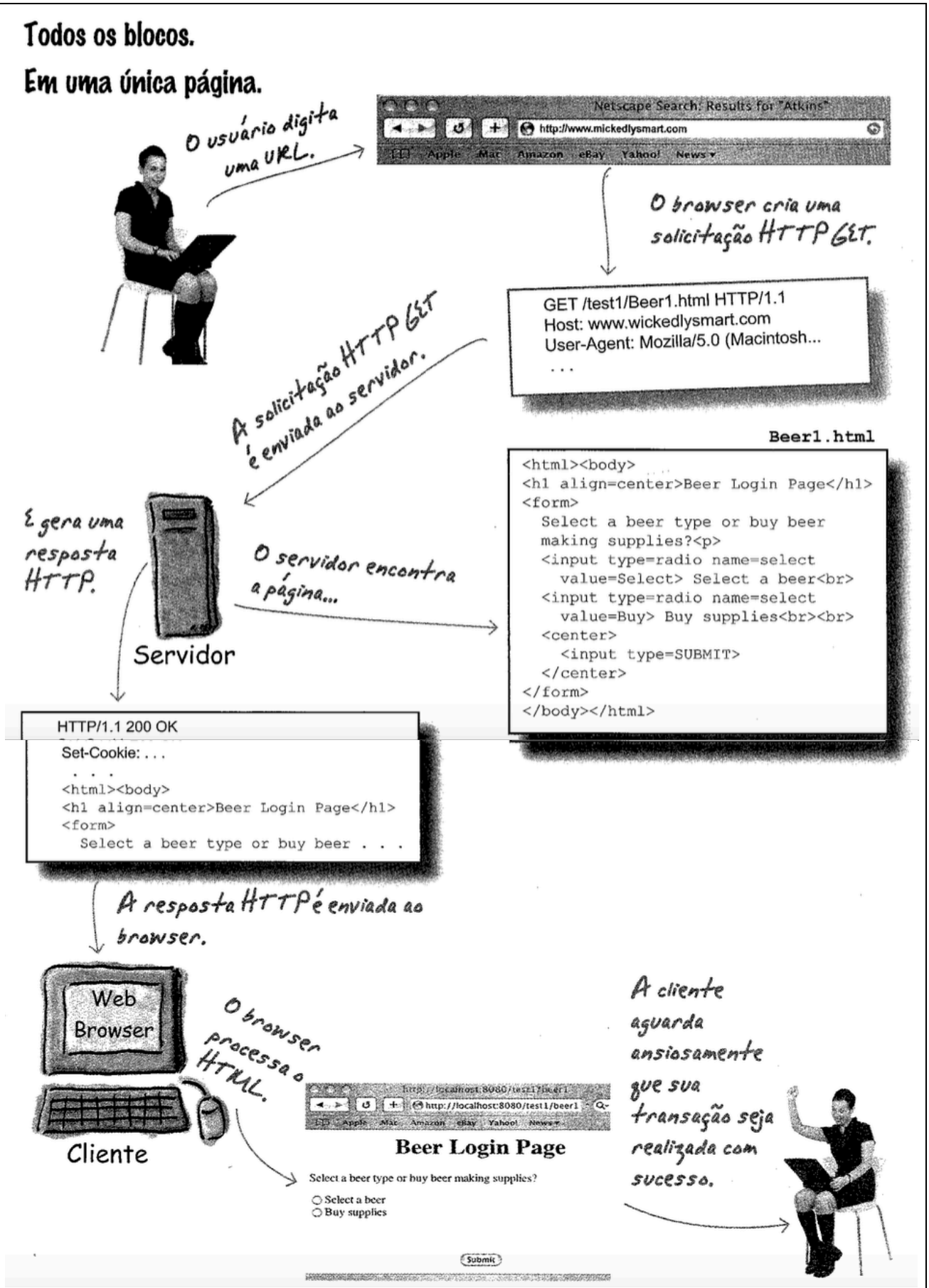
Fonte: (BASHAM, SIERRA, BATES 2008), p. 16

Resposta

A resposta é um pacote de dados HTTP, formatado como HTML. O conteúdo HTML vem dentro do protocolo HTTP.



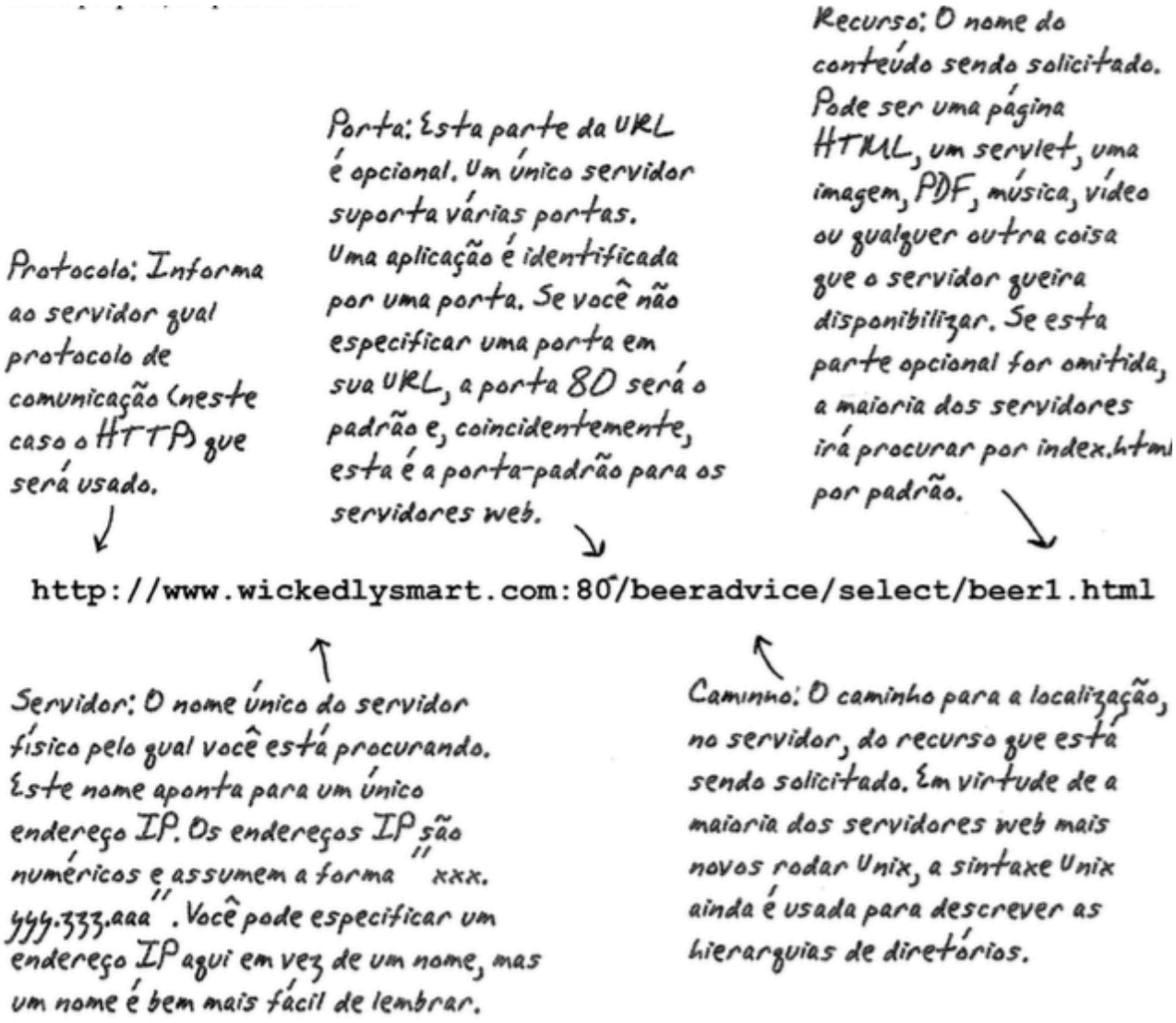
Fonte: (BASHAM, SIERRA, BATES 2008), p. 17



Fonte: (BASHAM, SIERRA, BATES 2008), p. 18

Uniform Resource Locator

É a URL, utilizada para a obtenção de recursos no servidor pelo navegador web.



Fonte: (BASHAM, SIERRA, BATES 2008), p. 20

Design Patterns

Conceitos Básicos

Padrões de Projeto (Design Patterns) são soluções usadas e experimentadas para problemas comuns do desenvolvimento de software. Os Design Patterns foram idealizados por Kent Beck, o mesmo da Extreme Programming, na década de 80. O primeiro pattern documentado do qual se tem notícia é o MVC, publicado em 88 por Karner & Pope no Journal of OOP como um padrão para

a organização do código em SmallTalk. Mas o assunto ganhou fama mesmo quando Gamma, Helm, Johnson e Vlissides, a Gang of Four (GoF), publicou um livro com 23 design patterns, inspirando-se no trabalho do arquiteto (de prédios, não de software) Christopher Alexander.

Vocês já tiveram uma visão de patterns ao aprenderem em METDS, no ano passado, os patterns GRASP, de Craig Larman.

Os patterns devem ser descritos de uma maneira clara e organizada para facilitar seu uso. A GoF sugere o seguinte modelo, que não é obrigatório. Há outros modelos de descrição de patterns, mas todos contém mais ou menos as mesmas informações.

Modelo para Descrição de Patterns

Nome do Pattern: o nome deve indicar a essência do pattern.

Classificação: quanto ao propósito, o GoF classifica os patterns como criacionais (lidam com criação de objetos), estruturais (tem a ver como os objetos se relacionam) e comportamentais (como os objetos interagem e como é feita a distribuição de responsabilidades); quando ao escopo, os patterns se classificam em de classe e de objetos.

Intenção: um texto curto que diz o que o pattern faz, explica sua razão de ser e sua intenção e, principalmente, qual o problema que ele pretende resolver.

Apelido: outros nomes pelos quais o pattern é conhecido.

Motivação: um cenário que ilustra um problema de design e como as classes e objetos se estruturam no pattern para resolver o problema.

Aplicabilidade: as situações em que o pattern pode ser aplicado.

Estrutura: uma representação gráfica das classes e objetos no pattern, usando UML; sempre há o diagrama de classes e, geralmente, o de sequência.

Participantes: as classes e objetos que participam do design pattern e suas responsabilidades.

Colaborações: como os participantes colaboram na execução de suas responsabilidades.

Consequências: quais são as escolhas que devem ser feitas para o uso deste pattern.

Implementação: armadilhas, dicas e técnicas para o implementador.

Código exemplo: fragmentos de código mostrando o uso do pattern em C++ ou Java. Usos conhecidos: exemplos dos patterns encontrados em sistemas reais.

Patterns relacionados: quais patterns estão relacionados com este, mostrando semelhanças e diferenças.

Alguns patterns bastante famosos são o Singleton e o Facade, e eles deram origem a diversos outros patterns aplicados a diversas situações. Para nossa disciplina uma das importantes utilizações é a aplicada à arquitetura JEE, publicada pela primeira vez em 2001 por Alur, Crupi e Malks. Estes patterns vem sendo disponibilizados gratuitamente pela Oracle em um catálogo de Patterns JEE que, a propósito, nunca fica pronto e muitos deles já estão desatualizados. Porém alguns são muito úteis ainda e iremos, nas próximas aulas, estudar os patterns Factory, DAO (Data Access Objects) e o TO (Transfer Object).

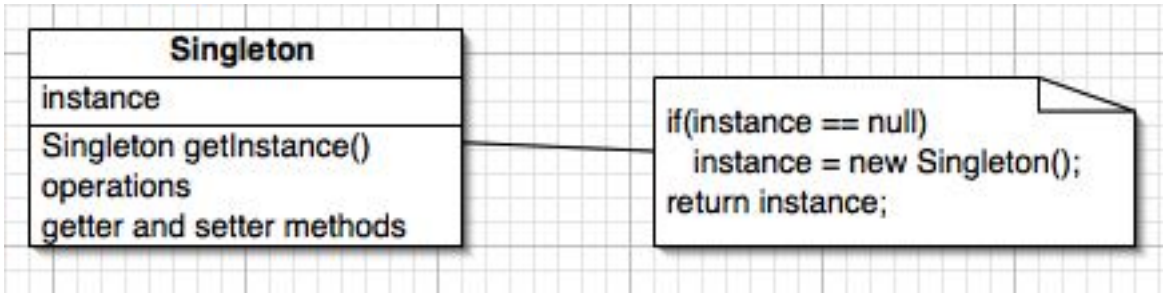
Exemplo de Pattern: Singleton

Intenção
– Garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso à mesma.

Motivação
– Em muitas situações é necessário garantir que algumas classes tenham uma e somente uma instância. Exemplo: o gerenciador de arquivos num sistema deve ser único.

Aplicabilidade
– Quando deva existir apenas uma instância de uma classe e essa instância deve dar acesso aos clientes através de um ponto bem conhecido.

Estrutura



Código Exemplo

```
public class ClassicSingleton {
        private static ClassicSingleton instance = null;
        private ClassicSingleton() {
        // Evita a instanciação por outra classe
        }
        public static ClassicSingleton getInstance() {
                if(instance == null) {
                        instance = new ClassicSingleton();
                }
        return instance;
        }
}
```

Colaborações
– Os clientes acessam uma única instância do Singleton pela operação getInstance()

Conseqüências
– Acesso controlado à instância única.

Padrões Correlatos
– AbstractFactory, Prototype, Builder

Bibliografia:

BASHAM, Bryan; SIERRA, Kathy; BATES, Bert. **Use a cabeça!: Servlets & JSP**. 2. ed. Rio de Janeiro: Alta Books, 2008-2010. xxxii, 879 p. ISBN 9788576082941 (broch.)

Quem é Christopher Alexander; disponível online em http://www.patternlanguage.com/ca/ca.html Acessado em 31/03/2015

KRASNER, G.; POPE, S.; **A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80**; Publicado originalmente em 1988 no Journal of Object Oriented Programming; disponível online em https://www.lri.fr/~mbl/ENS/FONDIHM/2013/papers/Krasner-JOOP88.pdf Acessado em 31/03/2015

Oracle; **Catálogo de Patterns JEE**; disponível online em http://www.oracle.com/technetwork/java/catalog-137601.html. Acessado em 21/02/2016

GAMMA, Erich. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos.** . Porto Alegre: Bookman, 2000. 364 p. ISBN 8573076100

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao processo unificado**. 3. ed. Porto Alegre: Bookman, 2007. 695 p. ISBN 9788560031528 (broch.)