

INE5404

+Interface Gráfica e Serialização

Prof. Jônata Tyska
Prof. Mateus Grellert



**UNIVERSIDADE FEDERAL
DE SANTA CATARINA**

Parte 4:

Usabilidade em

Interfaces

Gráficas

User Interface (UI) e User Experience (UX)

- **User Interface (UI):** lida com a criação de diversos protótipos com baixa e alta fidelidade da interface, bem como elementos de UI que os implementam
- **User Experience (UX):** lida com a experiência do usuário ao utilizar a aplicação. Preocupa-se com mecanismos de **interatividade**, com **satisfação do usuário** e com testes de usabilidade

User Experience (UX)

- Métrica qualitativa e de extrema importância
- Não existe receita universal
- Cada empresa pode possuir seu próprio conjunto de regras
 - Apple:
<https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>
 - Google: <https://material.io>

Heurísticas de Nielsen

- Para facilitar o trabalho de designers, Jakob Nielsen escreveu um artigo contendo 10 heurísticas para design de UI com foco em **Usabilidade**
- Chamam-se heurísticas, pois são **tentativas com base em conhecimento de domínio**, mas não devem ser enxergadas como regras absolutas
- Vamos ver cada uma com exemplos



Jakob
Nielsen

#1 – Visibilidade do Status do Sistema

- “O sistema deve sempre manter usuários informados sobre o que está acontecendo, com feedback apropriado em tempo razoável”
- Permite que o usuário se sinta no controle
- Pode ser persistente ou baseada em feedback

● Exemplos:

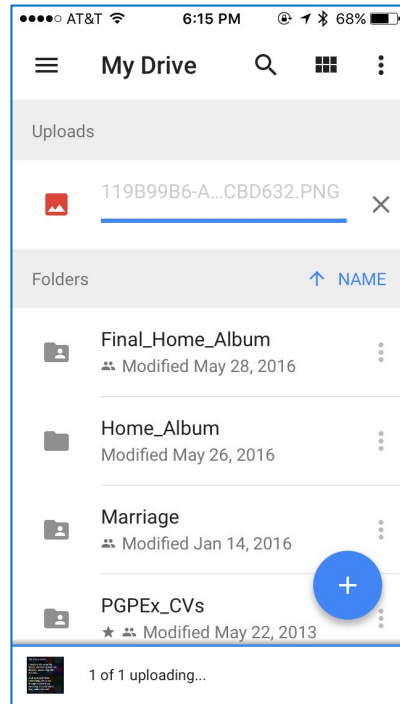
- Pressionar o botão de um elevador → **uma luz é acesa, indicando que o pedido foi registrado**
- A barra de status em celulares indica o **status de bateria e do sinal de WiFi**

- Evite **informações desnecessárias** para os usuários

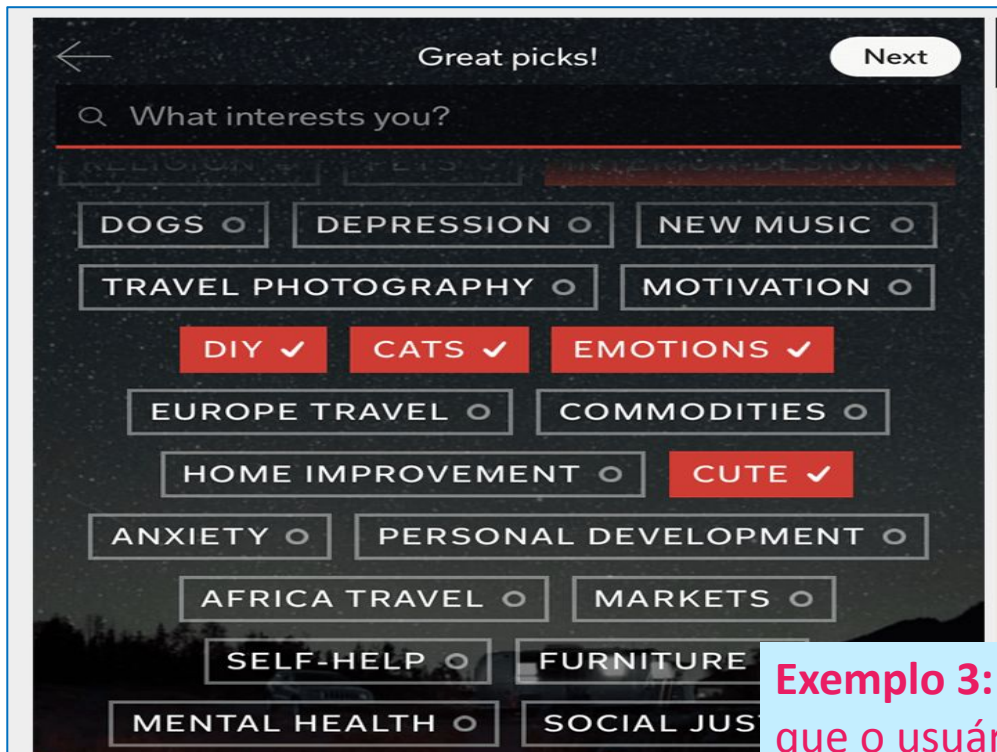
#1 – Visibilidade do Status do Sistema

- **Exemplo 1: barra de status de upload**

- A mesma interface mostra as pastas ordenadas por nome (definido pelo usuário)
- A parte inferior indica o progresso em termos de número de arquivos



#1 – Visibilidade do Status do Sistema



Exemplo 3: Destaque na cor a medida que o usuário seleciona as categorias

#2 - Compatibilidade entre o Sistema e o Mundo Real

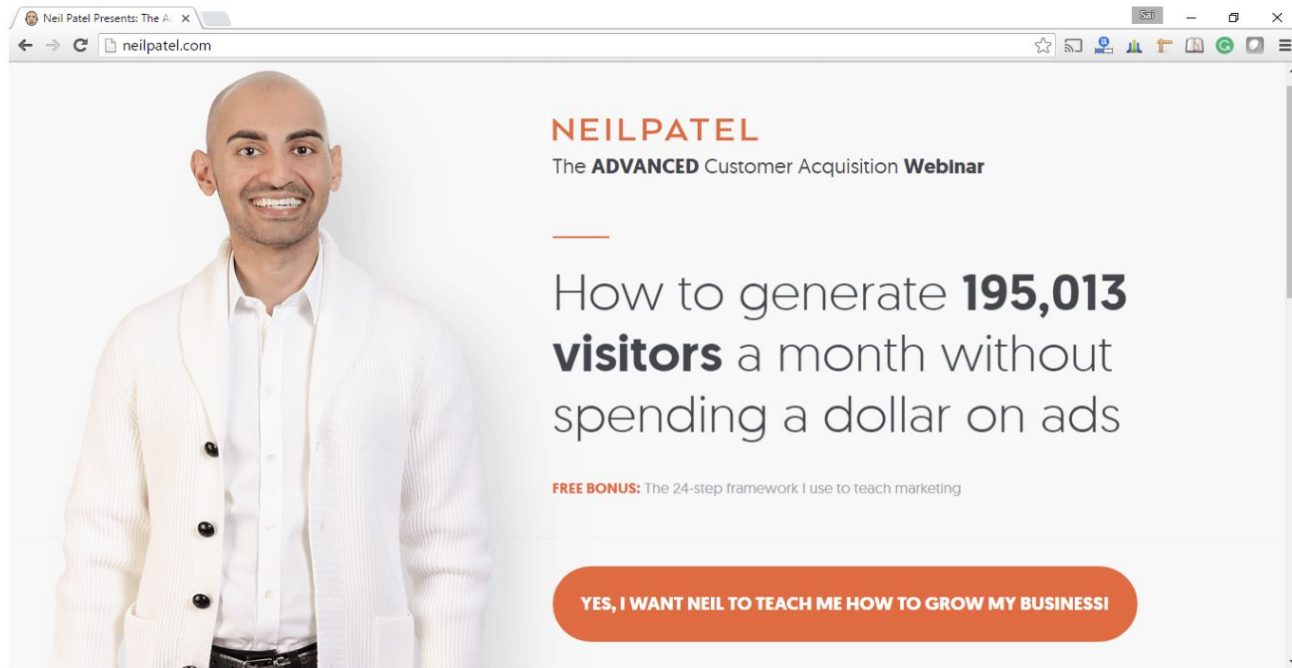
- “O sistema **deve falar a língua dos usuários**, com palavras e conceitos familiares, e **não termos técnicos** de desenvolvedores. Siga convenções do mundo real, fazendo com que a informação apareça de forma natural e intuitiva”
- Evitar termos confusos/ambíguos: “Conheça e Converse” x “Fale Conosco”
- **Skeuomorphic Design** (Esqueumorfismo): design que imita a aparência do objeto original somente para manter a familiaridade (e.g.: Kindle)

#2 - Compatibilidade entre o Sistema e o Mundo Real



Exemplo 1: Aplicativo de bússola com design similar ao objeto original

#2 - Compatibilidade entre o Sistema e o Mundo Real

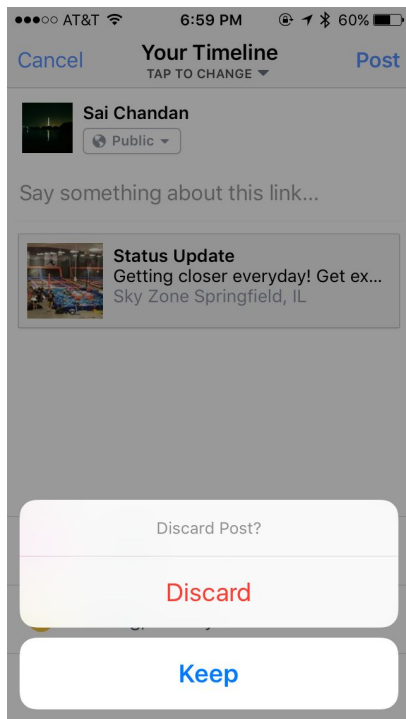


Exemplo 2: website com **botão explicando de forma clara** o que o cliente está comprando com o clique

#3 – Controle e Liberdade para o Usuário

- “Usuários frequentemente escolhem funções por engano e precisam de uma **saída de emergência** claramente marcada para abandonar a situação indesejada sem ter que passar por muito trabalho. Forneça opções de **desfazer** e **refazer**.”
- Botões de voltar, avançar em browsers
- Botão UNDO em editores de texto

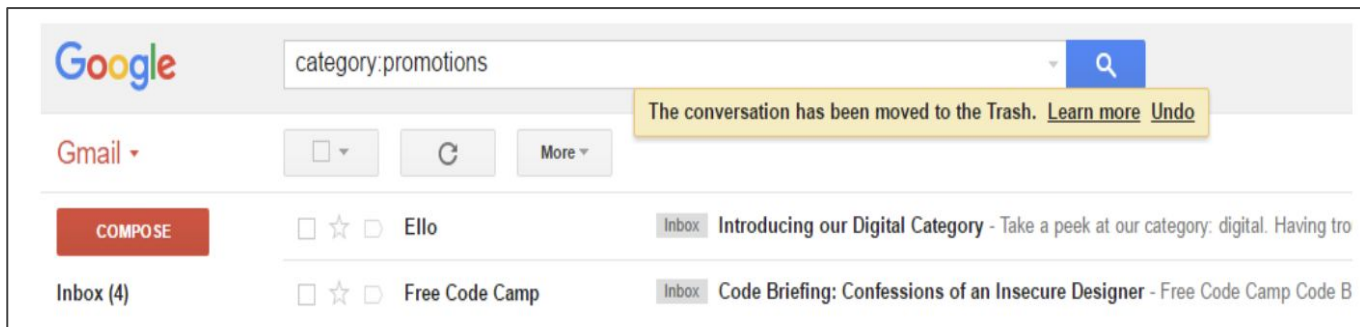
#3 – Controle e Liberdade para o Usuário



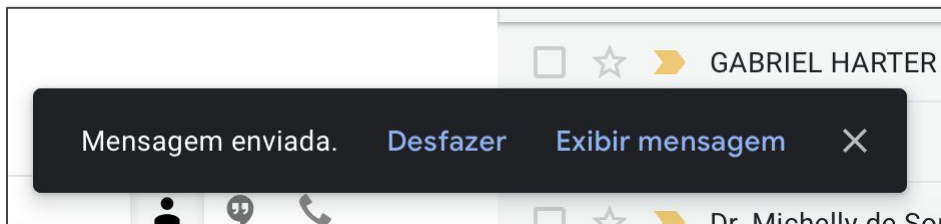
Exemplo 1: o mecanismo de **confirmação de exclusão** podem evitar que o usuário delete um arquivo por engano

#3 – Controle e Liberdade para o Usuário

Exemplo 2: mecanismo de desfazer uma exclusão do Gmail

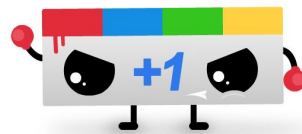


Exemplo 3: mecanismo de desfazer um envio do Gmail



#4 – Consistência e Padronização

- “Usuários não devem ter que pensar se a mesma coisa é dita com palavras, símbolos ou cores diferentes. Siga convenções de plataforma.”
- **Consistência interna:** manter o mesmo padrão em diferentes produtos da empresa → definir **Regras de Estilo** para a empresa
- **Consistência externa:** manter padrões que já foram consagrados (**exemplo:** botão Like do Facebook)



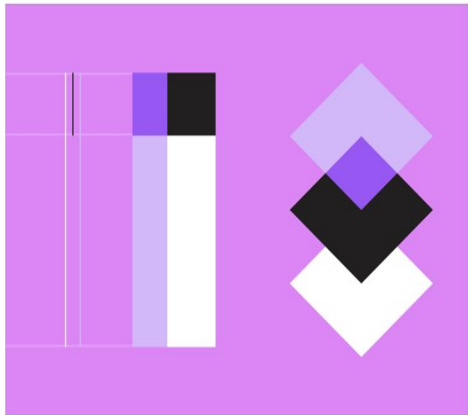
- **Lei de Jakob:** As pessoas passam a maior parte do seu tempo em sites que não são o seu

Fonte:

<http://www.brandflakesforbreakfast.com/2011/07/do-you-like-or-1.html>

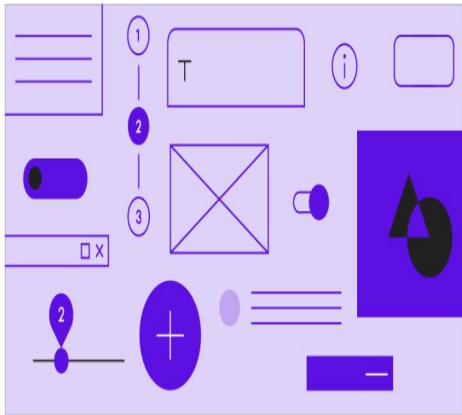
#4 – Consistência e Padronização

Exemplo 1: Material Design contém regras, ícones e outras ferramentas para um design padronizado e coerente



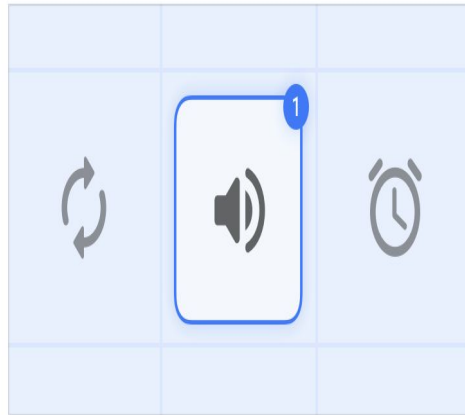
Material Design guidelines

Material Design principles, styles, and best practices



Components

Design guidance and developer documentation for interactive UI building blocks



Icons

Access five sets of stylized system icons, available in a range of formats and sizes

#4 – Consistência e Padronização



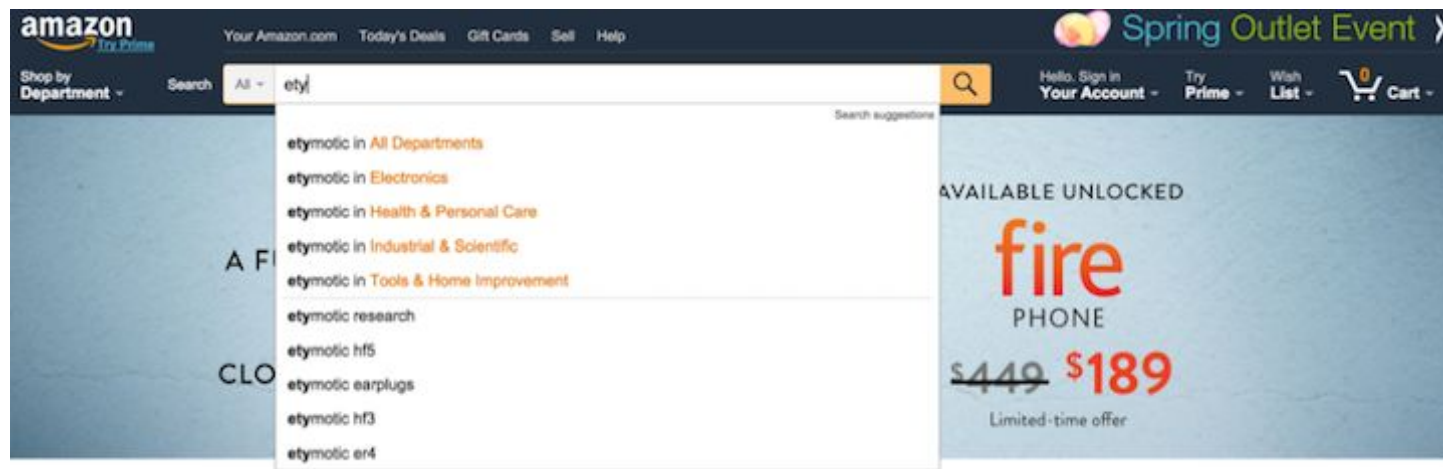
Exemplo 2: Sites de compra normalmente usam o carrinho de compras para representar a lista de compras

#5 – Prevenção de Erros

- “Ainda melhor do que boas mensagens de erro é um design cuidadoso que previne que o problema ocorra antes de qualquer coisa. Elimine condições suscetíveis a erro ou cheque esses erros e apresente aos usuários uma opção de confirmação antes que eles concretizem a ação.”
- **Deslizes:** Erro inconsciente. Acontece quando os usuários querem fazer uma coisa, mas fazem outra sem querer (erros de digitação).
- **Enganos:** Erro consciente. Acontece quando os usuários sabem o que querem e o que estão fazendo, mas a interface com o sistema ainda leva a uma experiência frustrada (Expectativa x Realidade) .

#5 – Prevenção de Erros

Exemplo 1: Mecanismo de sugestões em caixas de texto previnem erros de deslize na digitação do termo de busca



#5 – Prevenção de Erros

Exemplo 2: Controle de formatação Evita que caracteres inválidos sejam digitados previne uma série de inconvenientes para o usuário e para analista de dados

2 Profile

* NAME

First Name Last Name

* MOBILE NUMBER

+1 (555) 666-7778

* LANGUAGE

English (United States)

#5 – Prevenção de Erros

Exemplo 3: Detecção de força de senha reduz as chances de usuários escolherem uma senha pouco segura para autenticação.



Take it all with
Switch between devices, and pick up



Password strength: Too short

Use at least 8 characters. Don't use a password from another site, or something too obvious like your pet's name. [Why?](#)

Choose your username

@gmail.com

[I prefer to use my current email address](#)

Create a password

....|

Confirm your password

Birthdav

Confirmação de senha também reduz as chances de o usuário cadastrar uma senha errada.

#6 – Reconhecimento em vez de Memorização

- “Minimize a carga de memória do usuário criando objetos, ações e opções visíveis. O usuário não deve ter que lembrar da informação entre uma interação e outra. Instruções para uso do sistema devem ser visíveis ou facilmente recuperáveis quando apropriado.”
- Histórico de ações anteriores deve ser explicitado

#6 – Reconhecimento em vez de Memorização

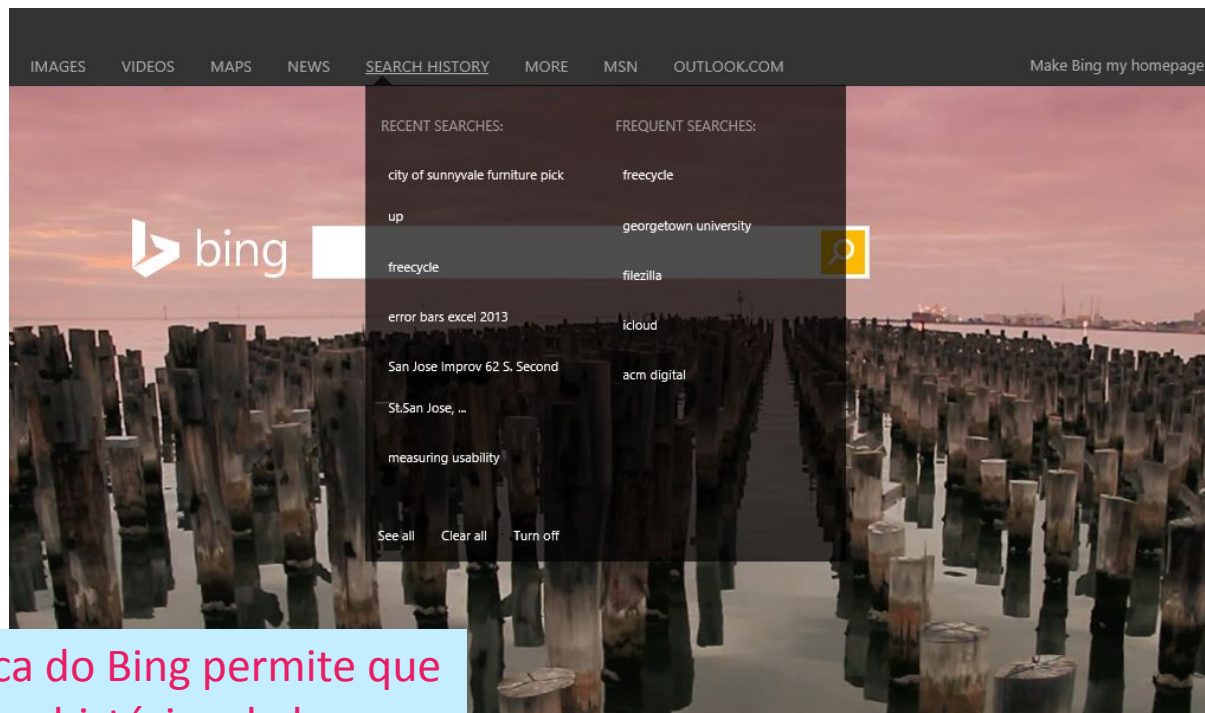
Exemplo 1: Lista de artigos visualizados recentemente na Amazon

Your Recently Viewed Items and Featured Recommendations

You
viewed



#6 – Reconhecimento em vez de Memorização



Exemplo 2: Sistema de busca do Bing permite que o usuário acesse facilmente o histórico de buscas


#7 – Flexibilidade de Facilidade no Uso

- “Aceleradores – não visíveis ao usuário iniciante – podem frequentemente acelerar/otimizar a interação para usuários experientes, de forma que ambos os tipos de usuários sejam atendidos. Permite que usuários tenham acesso fácil a ações frequentes.”
- Mensagens pré-prontas de e-mail
- Comandos CTRL+C/V/X/Z...

#7 – Flexibilidade de Facilidade no Uso

Account Setup
Enter Your Credentials

Exchange


Exchange

Sign in

Advanced Settings...

Exemplo 1: instalação avançada da ferramenta Office permite que usuários mais experientes removam ferramentas desnecessárias

#7 – Flexibilidade de Facilidade no Uso



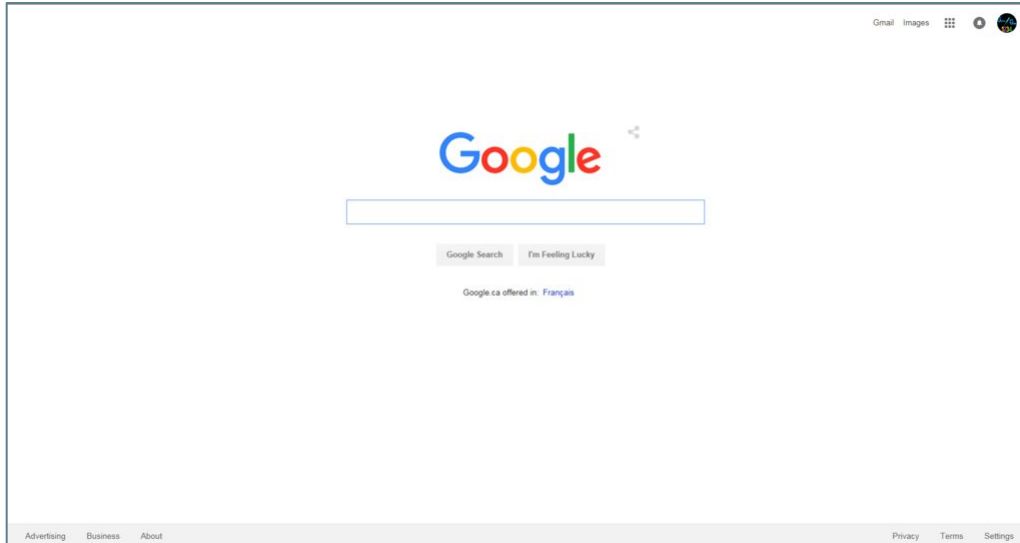
Exemplo 2: tap twice (aperte duas vezes) do aplicativo Instagram permite Like de posts de forma mais rápida

#8 – Estética e Design Minimalista

- “Interfaces não devem conter informação que seja irrelevante ou raramente necessária. Cada unidade extra de informação compete com as unidades relevantes e diminui sua visibilidade relativa.”
- **Comunicar** ao invés de **Decorar**

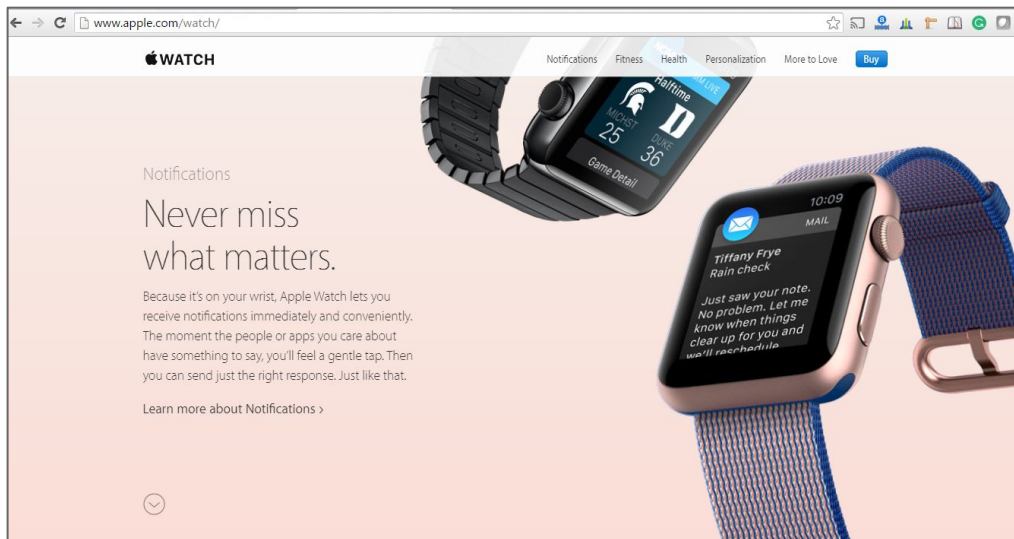
#8 – Estética e Design Minimalista

Exemplo 1: Página inicial site de buscas Google com design minimalista



#8 – Estética e Design Minimalista

Exemplo 2: Página do Apple Watch com destaque somente ao produto e ao botão de compra



#9 – Ajude os Usuários com os Erros

- “Mensagens de erro devem ser explicadas em linguagem simples, precisamente indicar o problema e sugerir uma solução de forma construtiva.”
- Exemplo **negativo**: **404 Not Found**

#9 – Ajude os Usuários com os Erros



INSCREVER-SE COM O FACEBOOK

☐ Eu concordo com os [Termos e condições do Spotify](#).

☐ Eu concordo com as [Condições de serviço](#) e dou consentimento para a coleta, o processamento e uso de meus dados pessoais, como descrito com mais detalhes na [Declaração de privacidade](#).

ou

Inscriver-se com seu endereço de e-mail

E-mail

Por favor, insira seu e-mail.

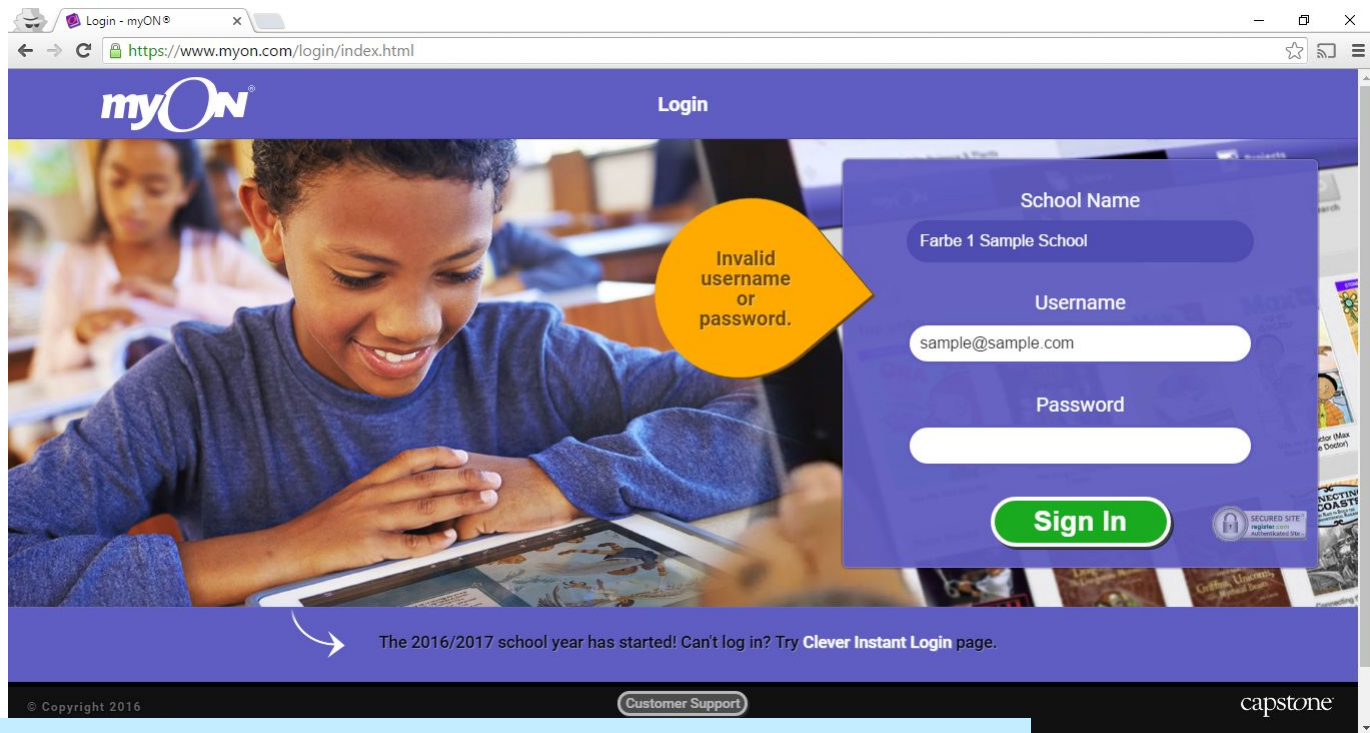
Confirmar e-mail

Por favor, insira seu e-mail.

Senha

Exemplo 1: Cadastramento no Spotify indicando como consertar os erros do form

#9 – Ajude os Usuários com os Erros



Exemplo 2: essa mensagem **não deixa claro** se o erro foi no usuário ou na senha (segurança?)

#9 – Ajude os Usuários com os Erros



⊗ Sorry, we couldn't find an account with that username. Can we help you recover your [username](#)?

Username

[I forgot](#)

freshsparkss

Password

[I forgot](#)

Show

Log In

☐ Stay logged in



⊗ Sorry, that password isn't right. We can help you [recover your password](#).

Username

[I forgot](#)

freshsparks

Password

[I forgot](#)

Show

Log In

☐ Stay logged in

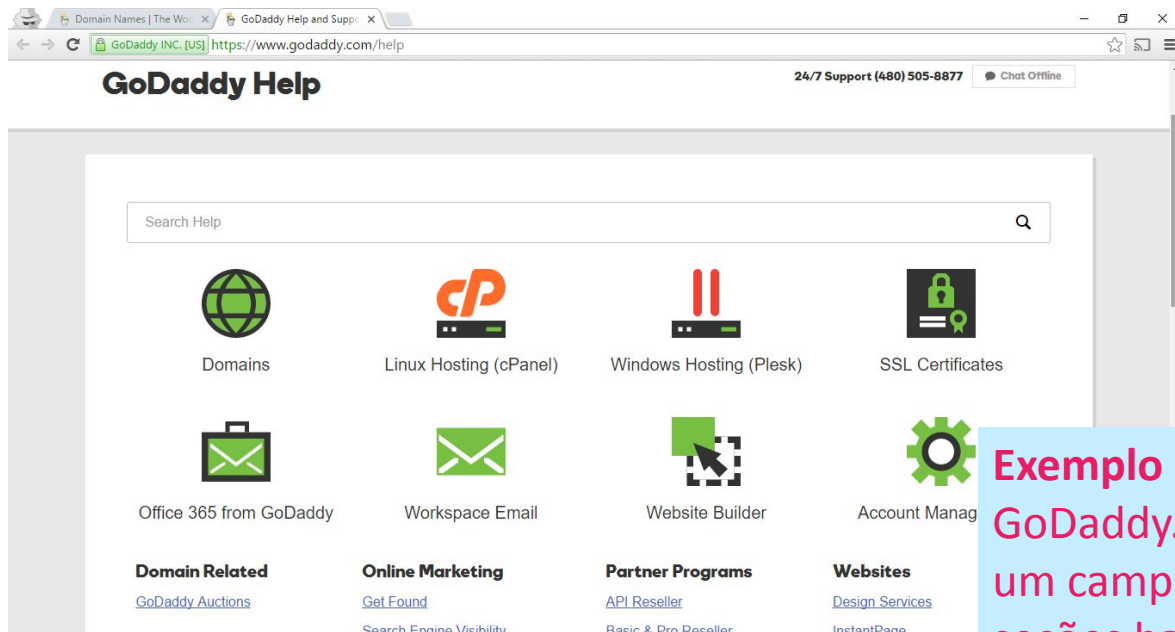
[Trouble logging in?](#)

Exemplo 3: mensagens mais específicas sobre o erro podem diminuir as frustrações de usuários no login

#10 – Ajuda e Documentação

- “Mesmo que seja melhor que o sistema seja utilizado sem documentação, pode ser necessário providenciar ajudar e documentação. Qualquer informação deve ser fácil de ser buscada, voltada à tarefa do usuário, listar os passos para resolver a questão, e não pode ser muito longa.”

#10 – Ajuda e Documentação



Exemplo 1: Centro de ajuda do site GoDaddy.com possui não somente um campo para pesquisa como seções baseadas em tópicos

Links com Vídeos Curtos sobre as 10 heurísticas

- <https://www.nngroup.com/videos/usability-heuristic-system-status/>
- <https://www.nngroup.com/videos/match-system-real-world/>
- <https://www.nngroup.com/videos/usability-heuristic-user-control-freedom/>
- <https://www.nngroup.com/videos/usability-heuristic-consistency-standards/>
- <https://www.nngroup.com/videos/usability-heuristic-error-prevention/>
- <https://www.nngroup.com/videos/recognition-vs-recall/>
- <https://www.nngroup.com/videos/flexibility-efficiency-use/>
- <https://www.nngroup.com/videos/aesthetic-and-minimalist-design/>
- <https://www.nngroup.com/videos/usability-heuristic-recognize-errors/>
- <https://www.nngroup.com/videos/help-and-documentation/>

**Até a
próxima!**

Parte 5:

Serialização

de objetos

Agradecimentos:
Prof. Jean Hauck

Persistência

- Nossos dados de programa normalmente ficam em uma região temporária da RAM (Heap + Stack), sendo eliminados pelo SO quando o programa termina
- O conceito de persistência surge justamente como a qualidade de um componente cujo estado sobrevive ao processo que o criou

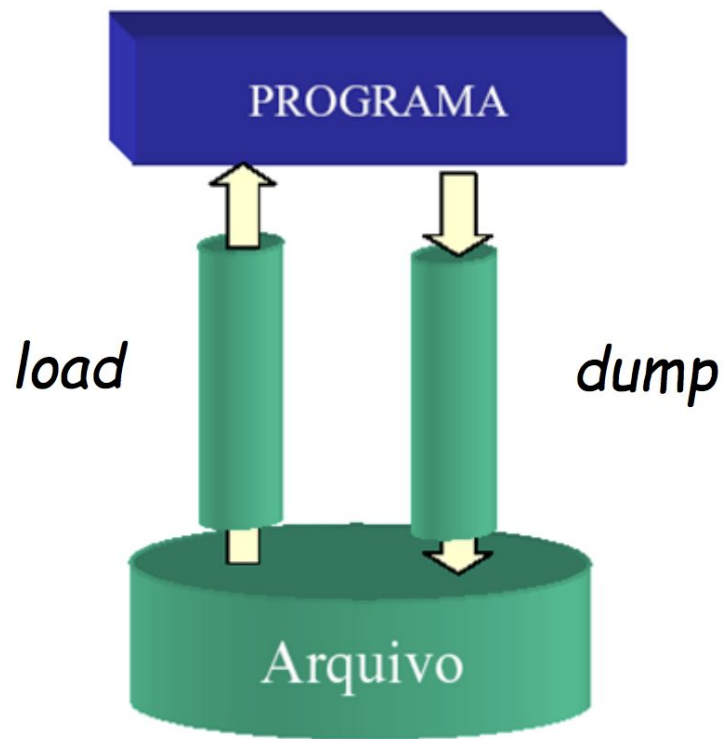
Persistência

Um requisito recorrente em desenvolvimento de software é a necessidade de que os dados **persistam** mesmo que o sistema não esteja executando:

- Recuperação em casos de erro
- Compartilhamento do estado interno de objetos

Persistência

- Uma forma simples de persistir dados é salvando-os em um dispositivo não volátil, como nosso HD
- No entanto, nossos objetos em Python não são arquivos de texto que podem facilmente ser escritos como uma string
- Para isso, utilizamos uma técnica chamada Serialização



Serialização

Duas operações:

1. Load
2. Dump

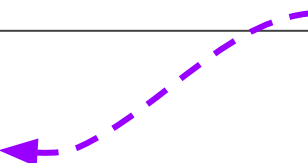
Serialização em Python

Em Python, podemos utilizar a biblioteca [pickle](#) para possível gravar **objetos serializáveis** diretamente em um arquivo **binário**:

- `import pickle`
- Serializamos os objetos com [dump](#)
- O método exige que um **arquivo binário de escrita** seja enviado como parâmetro

```
import pickle  
clientes = {}  
  
arq_clientes = open('clientes.pkl', 'wb')  
pickle.dump(clientes, arq_clientes)
```

Um dicionário que irá conter objetos da classe Cliente



Serialização em Python

Em Python, podemos utilizar a biblioteca [pickle](#) para possível gravar **objetos serializáveis** diretamente em um arquivo **binário**:

- `import pickle`
- Serializamos os objetos com [dump](#)
- O método exige que um **arquivo binário de escrita** seja enviado como parâmetro

```
import pickle  
clientes = {}  
arq_clientes = open('clientes.pkl', 'wb')  
pickle.dump(clientes, arq_clientes)
```

Nome do arquivo - é comum usamos **.pkl** para arquivos pickle



Serialização em Python

Em Python, podemos utilizar a biblioteca [pickle](#) para possível gravar **objetos serializáveis** diretamente em um arquivo **binário**:

- `import pickle`
- Serializamos os objetos com [dump](#)
- O método exige que um **arquivo binário de escrita** seja enviado como parâmetro

```
import pickle  
clientes = {}  
arq_clientes = open('clientes.pkl', 'wb')  
pickle.dump(clientes, arq_clientes)
```

w - modo escrita

b - modo binário



Serialização em Python

Em Python, podemos utilizar a biblioteca [pickle](#) para possível gravar **objetos serializáveis** diretamente em um arquivo **binário**:

- `import pickle`
- Serializamos os objetos com [dump](#)
- O método exige que um **arquivo binário de escrita** seja enviado como parâmetro

```
import pickle  
clientes = {}  
arq_clientes = open('clientes.pkl', 'wb')  
pickle.dump(clientes, arq_clientes)
```

Grava o objeto clientes no
arquivo arq_clientes

A purple callout box with white text is positioned to the right of the code block. A dashed purple line originates from the box and points to the `dump` method call in the last line of the code: `pickle.dump(clientes, arq_clientes)`.

Carregando arquivos serializados

Para ler os objetos serializados de um **arquivo binário**:

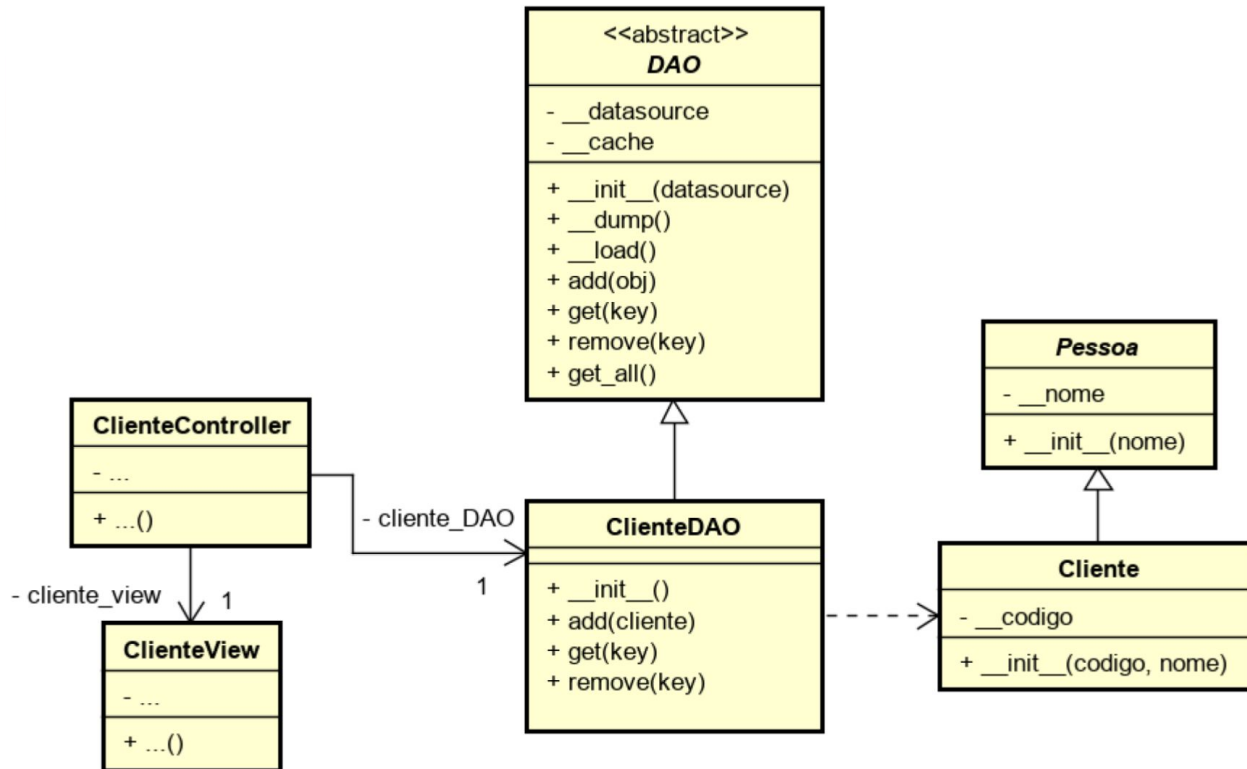
- Abrimos o arquivo como leitura (**read**), com tipo binário (**rb**)
- Carregamos os objetos com **load**

```
import pickle  
  
arq_clientes = open('clientes.pkl', 'rb')  
clientes = pickle.load(clientes, arq_clientes)
```

Criando classes persistentes

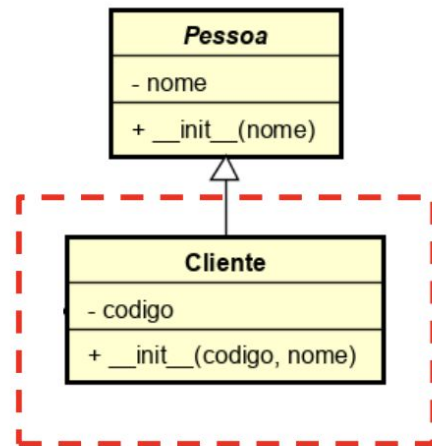
1. Definir quais classes serão persistentes
2. **Criar** uma classe para Acesso aos Dados da classe persistente
3. Implementar operação para **persistir o objeto**
4. Implementar operação para **carregar os objetos** do arquivo
5. Implementar **operações de recuperação e inserção** na lista controlada pela classe de Acesso aos Dados

Design Pattern: DAO – Data Access Object



Cliente - classe que terá persistência

```
class Cliente(Pessoa):  
  
    def __init__(self, codigo: int, nome: str):  
        super().__init__(nome)  
        self.__codigo = codigo  
  
    @property  
    def codigo(self):  
        return self.__codigo  
  
    @codigo.setter  
    def codigo(self, codigo):  
        self.__codigo = codigo
```



Classe DAO astrata

```
class DAO(ABC):
    def __init__(self, datasource=''):
        self.datasource = datasource
        self.objectCache = {}
        try:
            self.__load()
        except FileNotFoundError:
            self.__dump()

    def __dump(self):
        pickle.dump(self.objectCache, open(self.datasource, 'wb'))

    def __load(self):
        self.objectCache = pickle.load(open(self.datasource, 'rb'))
```

...

<<abstract>> DAO
- __datasource - __cache
+ __init__(datasource) + __dump() + __load() + add(obj) + get(key) + remove(key) + get_all()

Classe DAO astrata

```
class DAO(ABC):
    def __init__(self, datasource=''):
        self.datasource = datasource
        self.objectCache = {}
        try:
            self.__load()
        except FileNotFoundError:
            self.__dump()

    def __dump(self):
        pickle.dump(self.objectCache, open(self.datasource, 'wb'))

    def __load(self):
        self.objectCache = pickle.load(open(self.datasource, 'rb'))
```

...

<<abstract>> DAO
- __datasource - __cache
+ __init__(datasource) + __dump() + __load() + add(obj) + get(key) + remove(key) + get_all()

Classe DAO astrata

```
class DAO(ABC):  
    def __init__(self, datasource=''):  
        self.datasource = datasource  
        self.objectCache = {}  
        try:  
            self.__load()  
        except FileNotFoundError:  
            self.__dump()
```

```
    def __dump(self):  
        pickle.dump(self.objectCache, open(self.datasource, 'wb'))  
  
    def __load(self):  
        self.objectCache = pickle.load(open(self.datasource, 'rb'))
```

...

<<abstract>> DAO
- __datasource - __cache
+ __init__(datasource) + __dump() + __load() + add(obj) + get(key) + remove(key) + get_all()

Classe DAO astrata

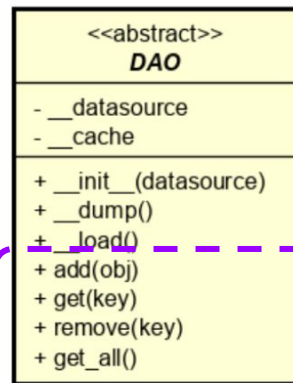
...

```
def add(self, key, obj):
    self.objectCache[key] = obj
    self.__dump()

def get(self, key):
    try:
        return self.objectCache[key]
    except KeyError:
        pass

def remove(self, key):
    try:
        self.objectCache.pop(key)
        self.__dump()
    except KeyError:
        pass

def get_all(self):
    return self.objectCache.values()
```



Classe DAO abstrata

...

```
def add(self, key, obj):  
    self.objectCache[key] = obj  
    self.__dump()
```

```
def get(self, key):  
    try:  
        return self.objectCache[key]  
    except KeyError:  
        pass
```

```
def remove(self, key):  
    try:  
        self.objectCache.pop(key)  
        self.__dump()  
    except KeyError:  
        pass
```

```
def get_all(self):  
    return self.objectCache.values()
```

<<abstract>> DAO	
-	__datasource
-	__cache
+	__init__(datasource)
+	__dump()
+	__load()
+	add(obj)
+	get(key)
+	remove(key)
+	get_all()

EAFP (Easier to Ask for Forgiveness than Permission) É mais “barato” deixar estourar a exceção do que todas as vezes procurar se existe

LBYL x EAFP

```
if hasattr(spam, 'eggs'):
    ham = spam.eggs
else:
    handle_error()
```

- Look before you leap (**LBYL**)
- focada em teste de **pré-condições**
- leva **mais tempo** se as exceções são raras

```
try:
    ham = spam.eggs
except AttributeError:
    handle_error()
```

- Easier to Ask for Forgiveness than Permission (**EAFP**)
- focada try/catch
- leva **mais tempo** se as exceções são comuns

Leia [aqui](#)

Classe ClienteDAO

```
class ClienteDAO(DAO):
    def __init__(self):
        super().__init__('clientes.pkl')

    def add(self, cliente: Cliente):
        if (isinstance(cliente.codigo, int)) and (cliente is not None) \
            and isinstance(cliente, Cliente):
            super().add(cliente.codigo, cliente)

    def get(self, key: int):
        if isinstance(key, int):
            return super().get(key)

    def remove(self, key: int):
        if isinstance(key, int):
            return super().remove(key)
```

ClienteDAO
+ __init__() + add(cliente) + get(key) + remove(key)

Classe ClienteDAO

```
class ClienteDAO(DAO):  
    def __init__(self):  
        super().__init__('clientes.pkl')
```

```
    def add(self, cliente: Cliente):  
        if (isinstance(cliente.codigo, int)) and (cliente is not None) \   
            and isinstance(cliente, Cliente):  
            super().add(cliente.codigo, cliente)
```

```
    def get(self, key: int):  
        if isinstance(key, int):  
            return super().get(key)
```

```
    def remove(self, key: int):  
        if isinstance(key, int):  
            return super().remove(key)
```

ClienteDAO
<ul style="list-style-type: none">+ __init__()+ add(cliente)+ get(key)+ remove(key)

Garante tipos de objetos e faz validações de negócio. Depois repassa objeto para "add" da Classe-Pai DAO

Faça você mesmo

Implemente essas classes e teste sua funcionalidade com o uso de pickle

Github com resultado

Serialização e Segurança

- Objetos serializados com pickle podem conter código malicioso
- Isso porque a binarização impede que a gente possa averiguar o conteúdo do objeto
- Uma alternativa para isso é utilizar uma serialização baseada em strings como JSON
 - Não pode executar código malicioso
 - É um padrão extremamente eficiente, utilizado por vários sistemas
 - Não consegue serializar todos os tipos Python (custom classes)

Leia mais [aqui](#)

**Até a
próxima!**