



Google App Engine

PUG-PE
março de 2010

Qualquer desenvolvedor pode **criar** e **hospedar** aplicações **web** usando a **infraestrutura** do **Google**.

A large white sign with the Google logo in its characteristic multi-colored font (blue, red, yellow, blue, green, red). The sign is mounted on a post and is positioned in the foreground of a modern building with large glass windows and a paved walkway. The background shows a well-maintained lawn and some shrubbery.

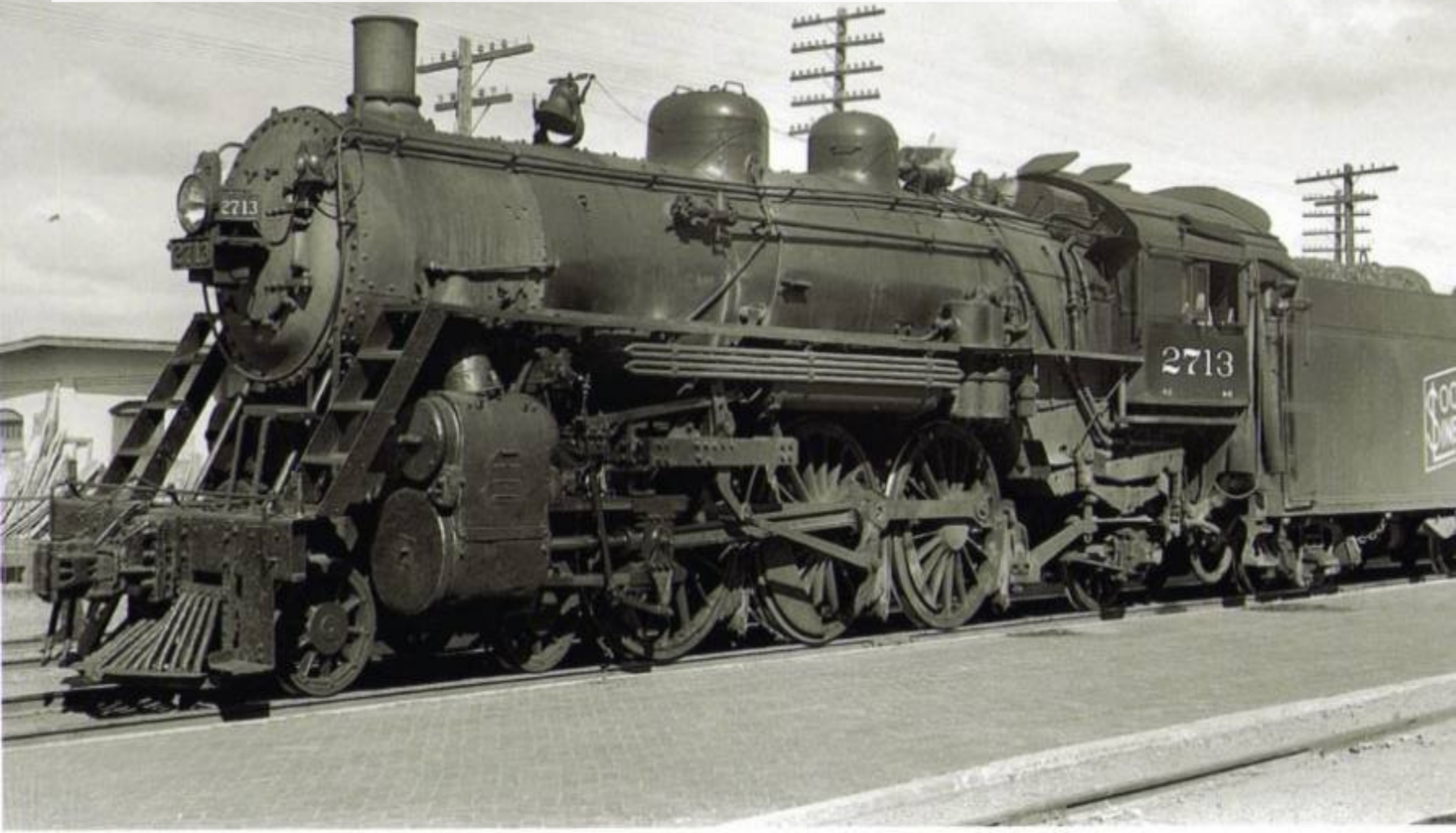
Google



B44

Inclusive **você**.

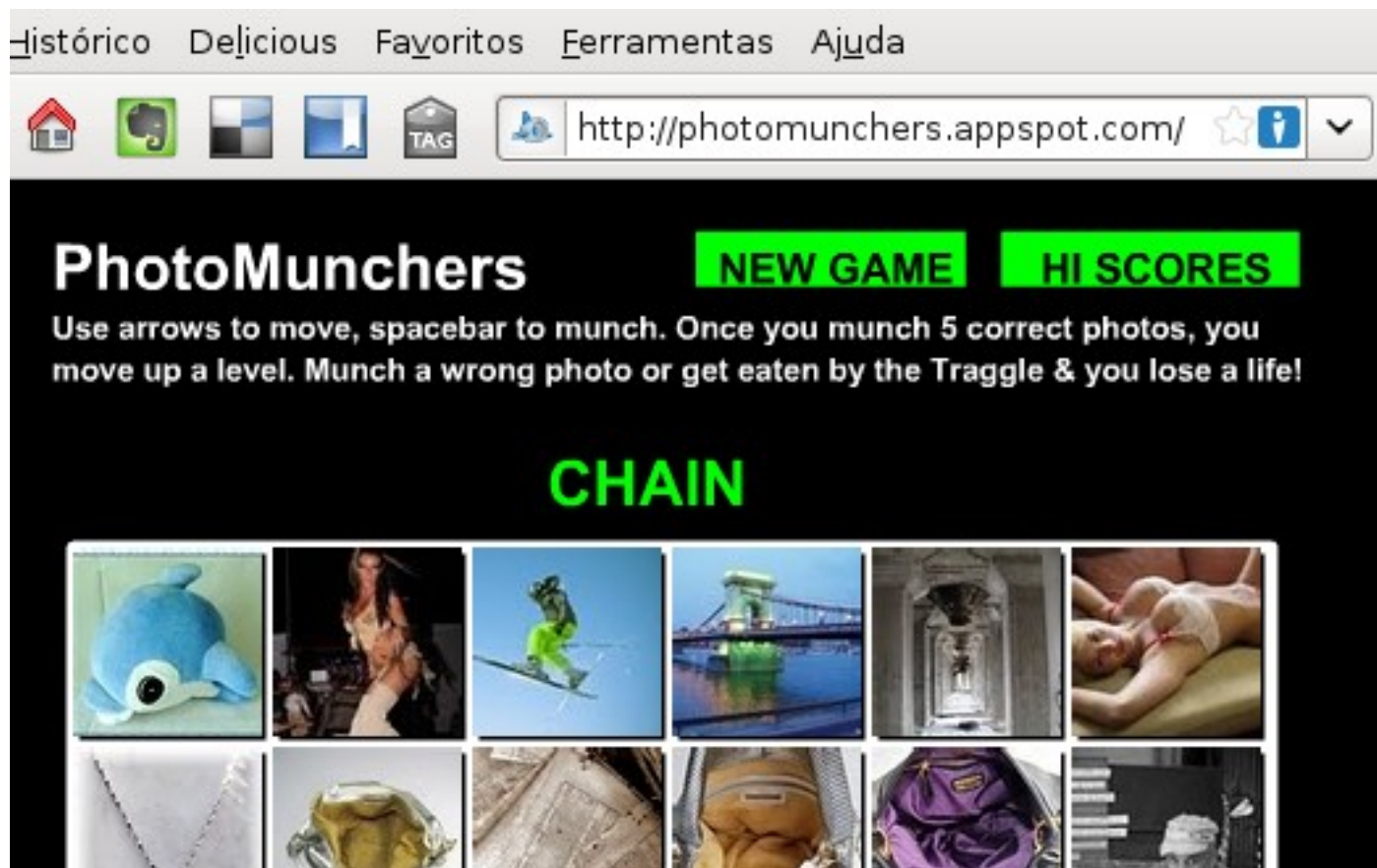
Isso significa ter mais **escalabilidade**,
disponibilidade e **desempenho**
em suas aplicações, mas ainda não é tudo...





A melhor parte é poder fazer
isso tudo usando **Python!**

É possível disponibilizar a aplicação em um domínio próprio ou como subdomínio de appspot.com



Armazenamento de Dados

- **Distribuído**, com suporte a **consultas** e **transações**
- Não relacional: **BigTable**
- Baseado em **entidades** que não possuem esquema
 - Estrutura é determinada pelo código da aplicação
 - Os objetos de dados possuem um tipo e um conjunto de propriedades

Autenticação de Usuários

- API usa as **contas do Google**, mas nada impede que se implemente um esquema próprio

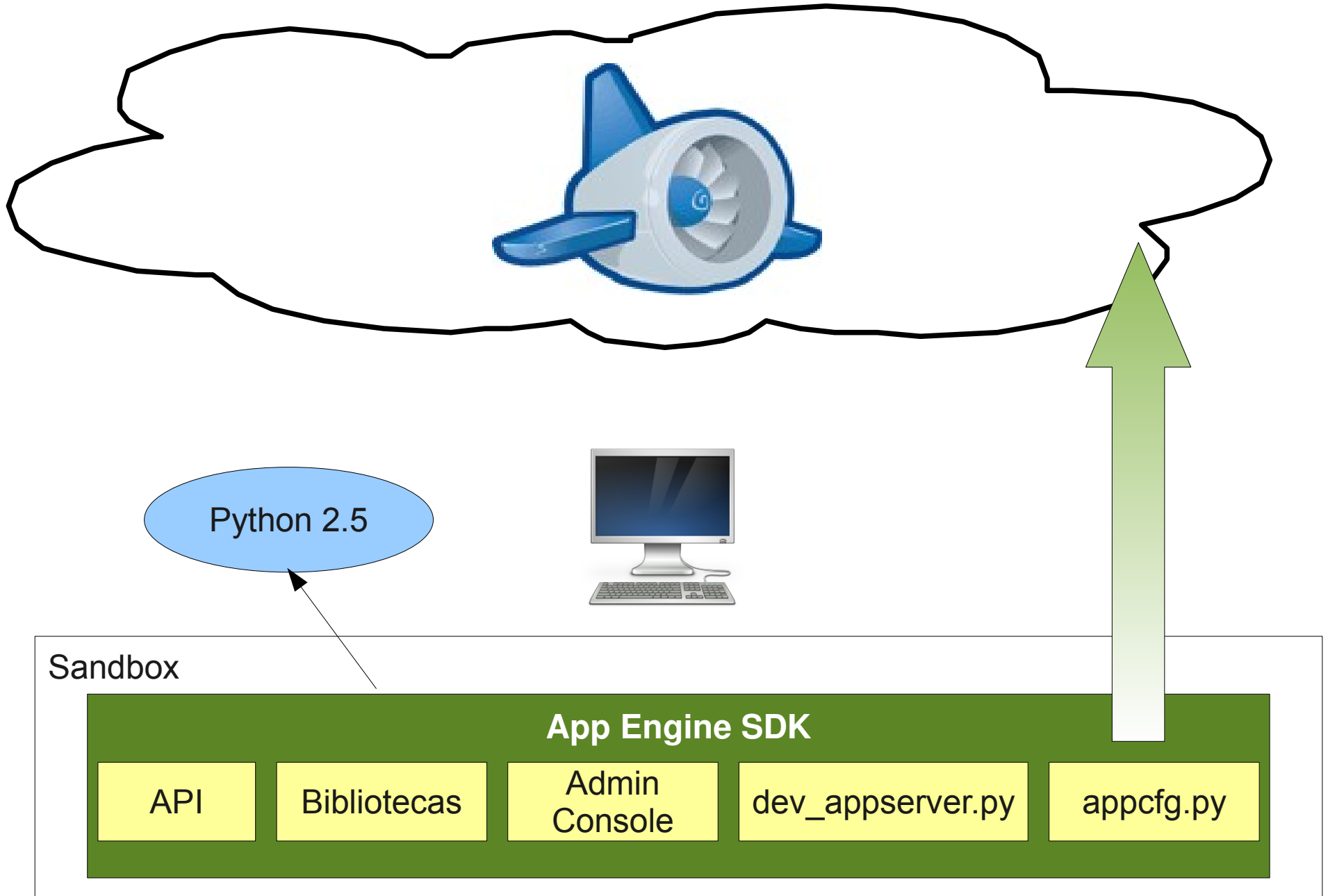
Outros Serviços

- URL fetching
- Envio de e-mails
- Memcache
- Manipulação de imagens
- Agendamento de tarefas

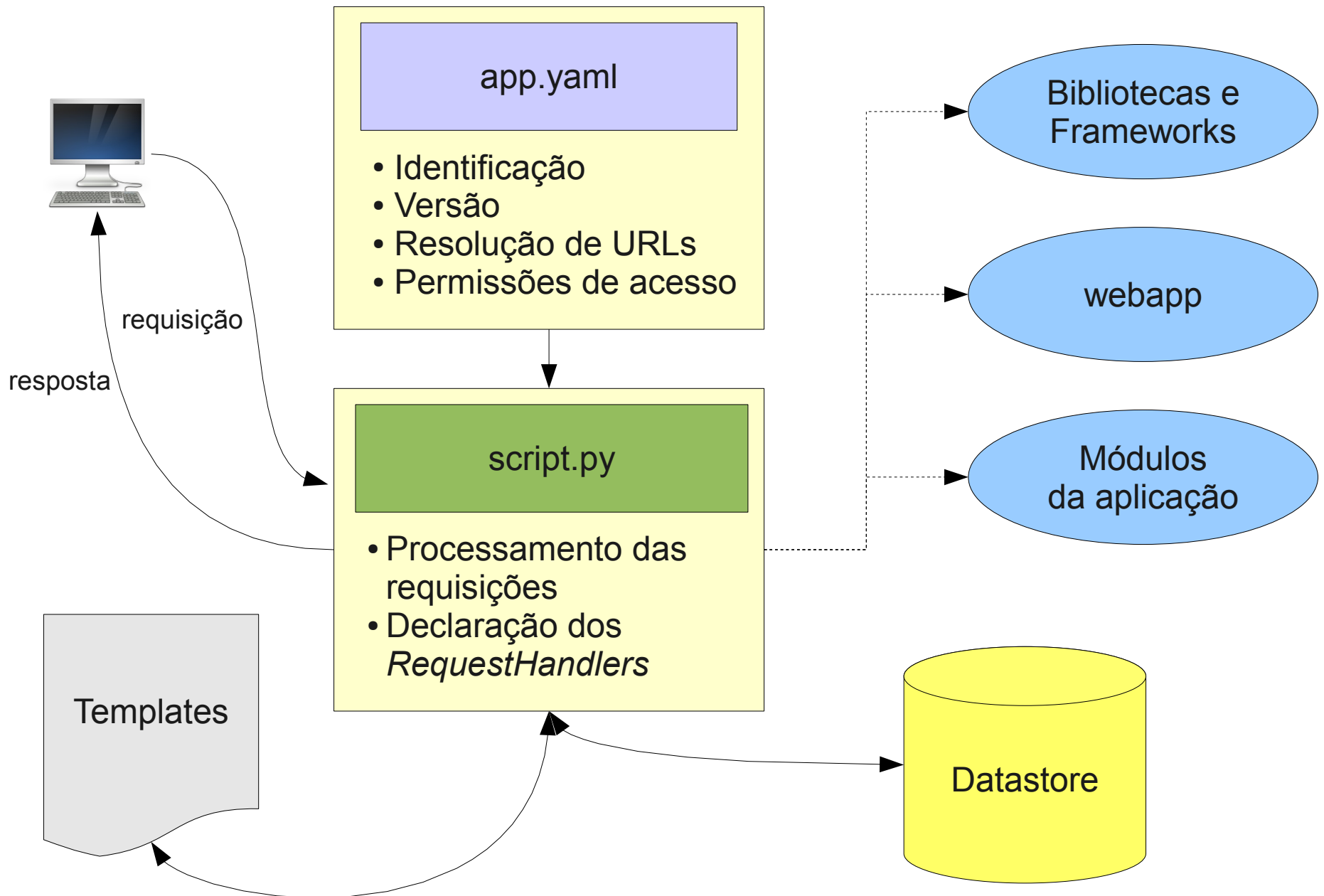
Algumas Restrições

- **10** aplicações por conta
 - **500 MB** de armazenamento
 - **5 milhões** de *pageviews* por mês
 - Requisição expira em **30 segundos** (*timeout*)
 - **6,5 horas de CPU** por dia
- ...além de outras quotas de uso por API (mail, urlfecth, XMPP etc.)

Fluxo de Desenvolvimento



Estrutura básica de uma aplicação



Componentes de uma aplicação

Um aplicativo **webapp** contém três partes:

- Classes **RequestHandler** que processam requisições e devolvem respostas
- Uma instância de **WSGIApplication** que redireciona as requisições recebidas para os **RequestHandlers**, com base no URL
- Uma rotina principal que executa o **WSGIApplication** usando um adaptador de CGI

A classe RequestHandler

Tem a finalidade de manipular uma requisição HTTP. Suas subclasses devem implementar um ou mais métodos correspondentes aos métodos de uma requisição HTTP:

- **get()**
- **post()**
- **put()**
- **options()**
- **delete()**
- **trace()**

A classe `WSGIApplication`

Representa um aplicativo que mapeia caminhos de URL para classes `RequestHandler`. Recebe como argumentos:

- `url_mapping`: uma lista de tuplas que mapeia cada uma URL da aplicação para o `RequestHandler` correspondente
- `debug`: valor `True` ou `False` que determina se a aplicação será executada em modo de depuração



Demonstração

Complementos e frameworks auxiliares

Alguns projetos que ajudam a reduzir a repetição de código e acelerar o desenvolvimento de aplicações:

- Google App Engine Helper for Django
 - ✓ <http://code.google.com/p/google-app-engine-django/>
- Google App Engine Oil
 - ✓ <http://code.google.com/p/google-app-engine-oil/>
- Google App Engine Patch (descontinuado)
 - ☒ <http://code.google.com/p/app-engine-patch/>

Referências

- Documentação do Google App Engine em Português
 - ✓ <http://code.google.com/intl/pt-BR/appengine/docs/python/>
- Video: “*Developing and deploying an application on Google App Engine*” (legendado)
 - ✓ <http://www.youtube.com/watch?v=bfgO-LXGpTM>
- Galeria de Aplicativos
 - ✓ <http://appgallery.appspot.com/>