UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CÂMPUS CORNÉLIO PROCÓPIO

ARQUITETURA DE SOFTWARE

MODELAGEM ARQUITETURAL E REST API

Augusto Henrique Salla Rosa

Lucas Reinaldi

Cornélio Procópio

2016

1. ARQUITETURA ESCOLHIDA

O padrão de arquitetura escolhido para que seja implementado o trabalho foi a Arquitetura de Cliente servidor mesclada a uma arquitetura de cadamas. Mesmo sendo um serviço simples, utilizar apenas uma arquitetura cliente-servidor poderia perder seu desempenho ao longo do tempo devido a quantidade de acessos a este serviço.

Alguns motivos pela escolha deste estilo foram suas vantagens, são elas:

- Permitir papeis e responsabilidades distribuidos;
- Facilidade de manutenção;
- Dados armazenados apenas no servidor;
- Segurança;
- Acesso a recursos;
- Atualizações;
- Distribuição;
- Funciona com vários clientes;

1.1 Os componentes do estilo arquitetural escolhido

Para esse estilo arquitetural de cliente-servidor, temos alguns pequenos modelos e componentes para que a lógica da estrutura se mantenha integra.

Temos o acesso de **multiplos clientes** em um **servidor** que aplica a usabilidade do padrão **REST.**

O servidor possuí duas **bases de dados** sendo um, a aplicabilidade do **Master e Slave**, onde serviços e consultas podem ser utilizados de melhor maneira. Garantindo transações e acesso aos recursos de banco.

1.2 Esquema gráfico geral da arquitetura escolhida

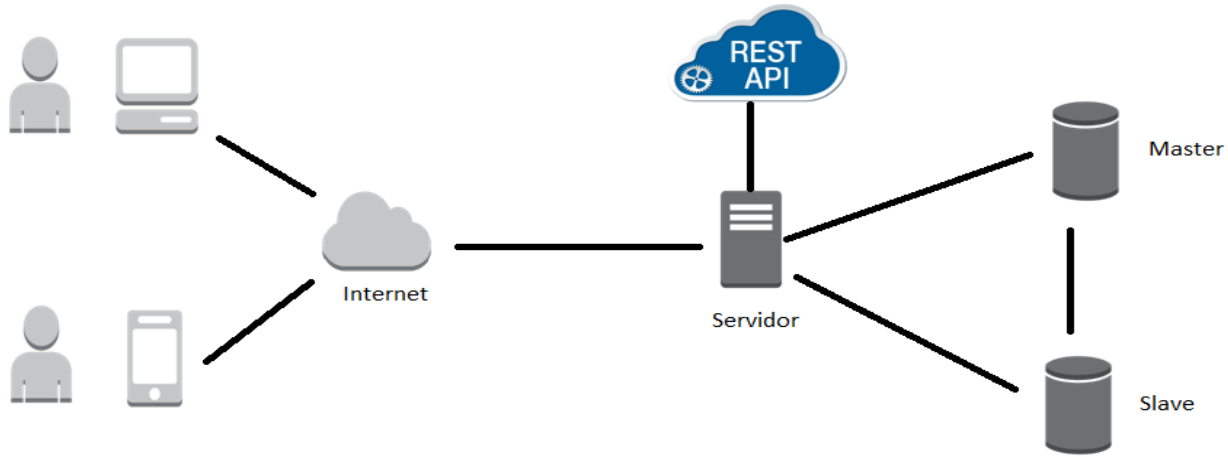

Figura 1: Esquema da representação gráfica da arquitetura de cliente servidor mesclada com camadas.
Fonte: Reinaldi, Lucas. Autoria Própria

1.2 Possibilidades de evoluções futuras da aplicação.

Poderia ser implementado formas de incluir mídias ao serviço, tais como fotos , audio, vídeos e formas de compartilhamento de conteúdo entre os que o utilizam. Tornando o sistema de tarefas mais interativo, dinâmico e apresentável.

Melhorar a arquitetura, implementando um sistema de controle de banco de dados mais perfomatico seria talvez uma implementação decente para o futuro.

Uma arquitetura Gatway seria muito bem aplicada também, melhorando assim o acesso a recursos rest do servidor.

Endpoint: https://api.tasky.io/

1.3 Mensagem padrão de erro

A mensagem padrão de erro será retornada um Json contendo o seguinte conteúdo:

Mensagem de erro de get Tarefas.

Caso id da consulta get /tasks/(id) seja vazia

‘error’ : [{“ERROR, Id cannot be NULL. Value is NULL”}];

Mensagem de erro putTasks.

Caso id do Put /tasks/(id) seja valor vazio

‘error’ : [{“ERROR, Id cannot be NULL. Value of is NULL”}];

Mensagem de erro PostCategorias

Caso a string titulo seja vazia do método POST /categories.

‘error’ :[{“ERROR, @String title cannot be NULL. Value of String is NULL”}];

2. RECURSOS UTILIZADOS DA API

A seguir será exemplificado por meio de tabelas a lista de recursos utilizados no trabalho. Incluindo o método HTTP, a descrição de cada recurso, os parêmetros utilizados (Obrigatórios e opcionais) e os exemplos de resposta utilizando o formato Json.

LEGENDA DE PARÂMETROS:

*** são obrigatórios**

**** são opcionais**

Tarefas (Tasks)

MÉTODO HTTP	DESCRIÇÃO	PARÂMETROS	RESPOSTAS
GET /tasks	Retorna todas as tarefas.	**category, *limit (paginação)	“listTasks”: [{ “category”: ”categoria1” } { “limit” : “10” }]
GET /tasks/(id)	Retorna/consulta a tarefa do id correspondente.	*id	“getTask”:[{“id” : “1”}, {“id” : ”2”}]
POST /tasks	Cria uma nova tarefa.	*title, *description, *date, **category, **reminder, *status	“inserTask”: [{“title” : “Tarefa 1”}, {“description” : “Desc. Tarefa11”}, {“date” : “00/00/0001”}, {“category” : “Categoria 1”}, {“reminder” : “Lembrete 1”}, {“status” : “Estado pronto 202”}]
PUT /tasks/(id)	Atualiza a tarefa de id correspondente.	*id, **title, **description, **date, **category, **status	“UpdateTasks”:[{“id” : “1”}, {“title” : “Tarefa 1”}, {“description” : “DescTarefa 1”}, {“date” : “01/01/0001”}, {“category” : “Categoria 1”}, {“status” : “error 404”}]
DELETE /tasks/(id)	Deleta a tarefa de id correspondente.	*id	“DeleteTask”: [{“id” : “1”}]

Categories (Categorias)

MÉTODO HTTP	DESCRIÇÃO	PARÂMETROS	RESPOSTAS
GET /categories	Retorna todas as categorias	*limit (paginação)	“listCategories”: [{ “limit” : “10” }]
GET /categories/(id)	Retorna/consulta a categoria do id correspondente	*id	“getCategories”: [{“id” : “1”}]
POST /categories	Cria uma nova categoria	*title, *description	“insertCategories”: [{“title” : “Categoria 1”}, {“description” : “DescCat1”}]
PUT /categories(id)	Atualiza a categoria de id correspondente	*id, **title, **description	“updateCategories”: [{“id” : “1”}, {“title” : “Categoria 1”}, {“description” : “DescCat1”}]
DELETE /categories/(id)	Deleta a categoria de id correspondente	*id	“deleteCategories”: [{“id” : “1”}]

Reminders (Lembretes)

MÉTODO HTTP	DESCRIÇÃO	PARÂMETROS	RESPOSTAS
GET/reminders	Retorna todos os lembretes	*limit (paginação)	“listReminders”: [{“limit” : “10”}]
GET /reminders/(id)	Retorna/Consulta o lembrete do id correspondente.	*id	“getReminders”: [{“id” : “1”}]
POST /reminders	Cria um novo lembrete.	*title, *decsription, **task	“insertReminders”: [{“title” : “Lembrete 1”}, {“description” : ”DescLem1”}, {“task” : “Tarefa 1”}]
PUT/reminders/(id)	Atualiza o lembrete de id correspondente.	*id, **title, **description, **task	“updateReminders”: [{“id” : “1”}, {“title” : “Lembrete 123”}, {“description” : “DescLem123”}, {“task” : “Tarefa 23”}]
DELETE /reminders/(id)	Deleta o lembrete de id correspondente	*id	“deleteReminders”: [{“id” : “1”}]