PHOENIX FRAMEWORK

WARUM PHOENIX?

FEATURES

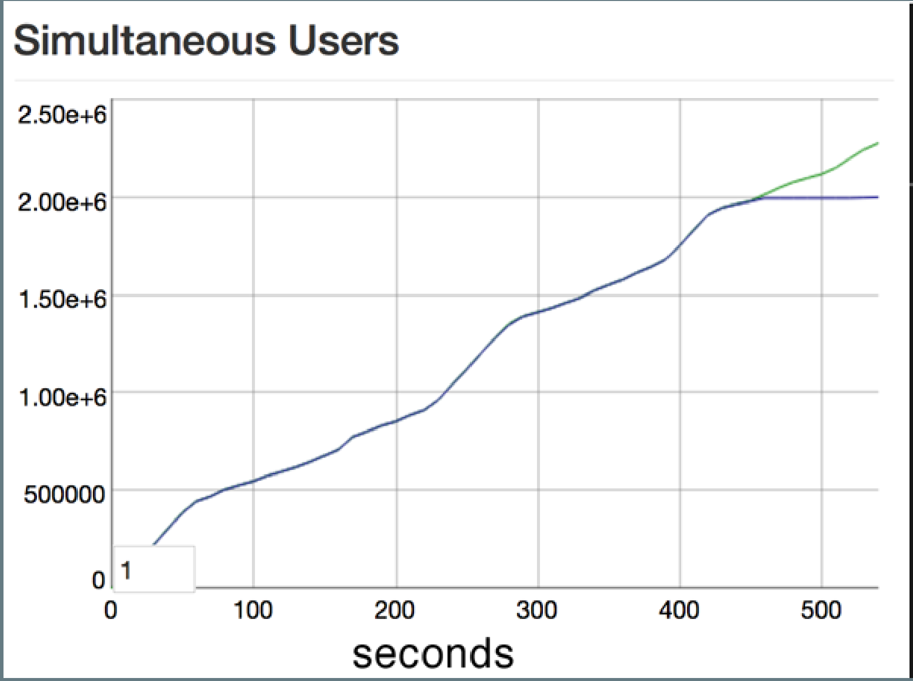
- Uptime/Fehlertoleranz
- Websockets
- Verhalten unter Last
- Transparenter Aufbau
- Wenig “Magie”
- Wartbarkeit

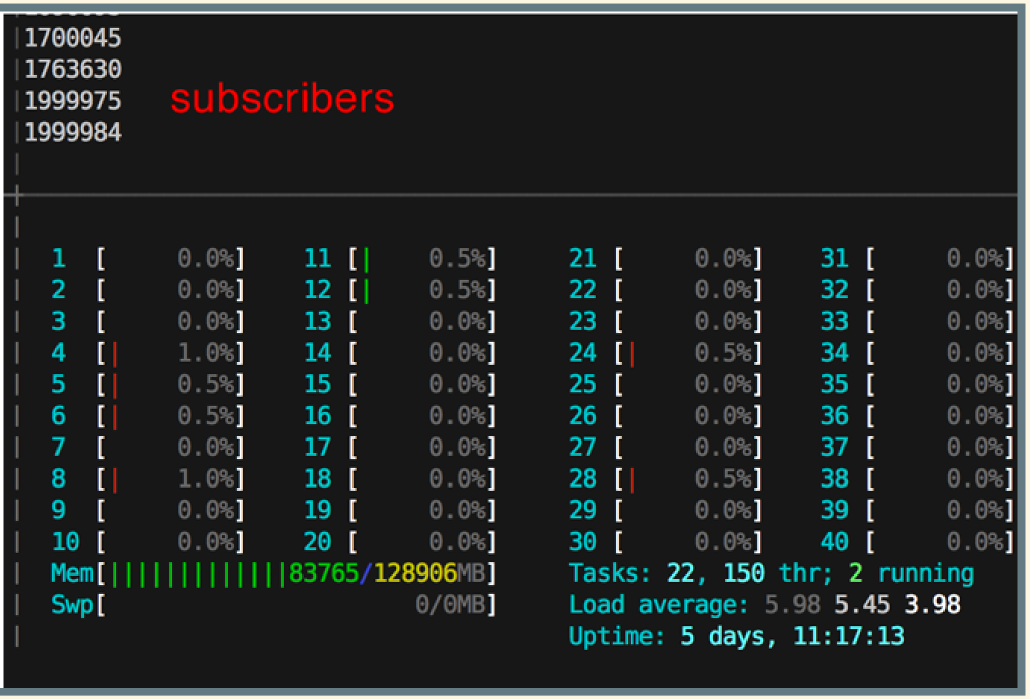
PERFORMANCE

Framework	Throughput (req/s)	Latency (ms)	Consistency (σ ms)
Plug	198328.21	0.63ms	2.22ms
Phoenix	179685.94	0.61ms	1.04ms
Gin	176156.41	0.65ms	0.57ms
Play	171236.03	1.89ms	14.17ms
Express Cluster	92064.94	1.24ms	1.07ms
Martini	32077.24	3.35ms	2.52ms
Sinatra	30561.95	3.50ms	2.53ms
Rails	11903.48	8.50ms	4.07ms

https://gist.github.com/omnibs/e5e72b31e6bd25caf39a

PERFORMANCE

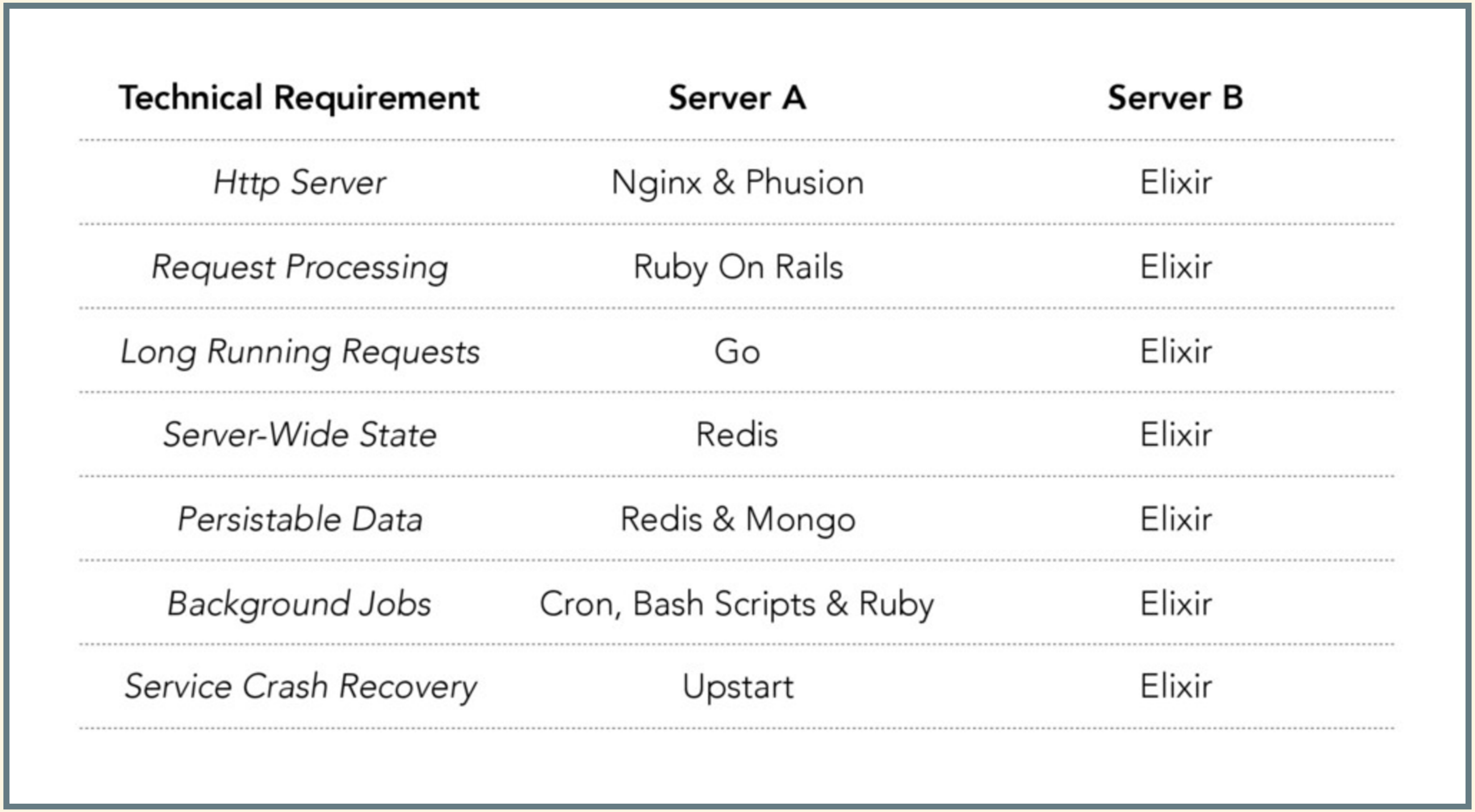




```
|1700045
|1763630
|1999975
|1999984
|
+
|
|  1  [        0.0%]    11 [|       0.5%]    21 [        0.0%]    31 [        0.0%]
|  2  [        0.0%]    12 [|       0.5%]    22 [        0.0%]    32 [        0.0%]
|  3  [        0.0%]    13 [        0.0%]    23 [        0.0%]    33 [        0.0%]
|  4  [|       1.0%]    14 [        0.0%]    24 [|       0.5%]    34 [        0.0%]
|  5  [|       0.5%]    15 [        0.0%]    25 [        0.0%]    35 [        0.0%]
|  6  [|       0.5%]    16 [        0.0%]    26 [        0.0%]    36 [        0.0%]
|  7  [        0.0%]    17 [        0.0%]    27 [        0.0%]    37 [        0.0%]
|  8  [|       1.0%]    18 [        0.0%]    28 [|       0.5%]    38 [        0.0%]
|  9  [        0.0%]    19 [        0.0%]    29 [        0.0%]    39 [        0.0%]
|  10 [        0.0%]    20 [        0.0%]    30 [        0.0%]    40 [        0.0%]
|  Mem[||||||||||||83765/128906MB]          Tasks: 22, 150 thr; 2 running
|  Swp[                    0/0MB]          Load average: 5.98 5.45 3.98
|                                           Uptime: 5 days, 11:17:13
```

http://www.phoenixframework.org/blog/the-road-to-2-million-websocket-connections

STACK



Technical Requirement	Server A	Server B
Http Server	Nginx & Phusion	Elixir
Request Processing	Ruby On Rails	Elixir
Long Running Requests	Go	Elixir
Server-Wide State	Redis	Elixir
Persistable Data	Redis & Mongo	Elixir
Background Jobs	Cron, Bash Scripts & Ruby	Elixir
Service Crash Recovery	Upstart	Elixir

https://speakerdeck.com/bcardarella/from-rails-to-phoenix?slide=71

AUFBAU

ERSTELLEN EINES PROJEKTES

```
$ mix phoenix.new myapp
```

VERZEICHNISSTRUKTUR

```
$ tree -dL 2 myapp
myapp
├── config
├── lib
│   └── myapp
├── priv
├── test
└── web
    ├── channels
    ├── controllers
    ├── models
    ├── static
    ├── templates
    └── views
```

LAYERS OF PHOENIX

```
connection
|> endpoint
|> router
|> pipelines
|> controller
```

McCord, Chris et al. *Programming Phoenix* (1st ed.), p. 17

CONNECTION

```
%Plug.Conn{
  method: "GET",
  request_path: "/",
  req_headers: [...],
  params: %{...},
  cookies: %{...},
  assigns: %{...},
  resp_body: "Hello, world!",
  resp_headers: [...],
  ...
}
```

ACTION

```
connection
|> find_user
|> view
|> template
```

McCord, Chris et al. *Programming Phoenix* (1st ed.), p. 18

GRUNDLAGEN

SCAFFOLDING

- Generieren eines Grundgerüsts für eine Ressource
- CRUD (Create Read Update Delete)

```
$ mix phoenix.gen.html Post posts title:string body:text
```

MIGRATION

```elixir
defmodule Myapp.Repo.Migrations.CreatePost do
  use Ecto.Migration

  def change do
    create table(:posts) do
      add :title, :string
      add :body, :text
      timestamps()
    end
  end
end
```

MODEL/SCHEMA

```elixir
defmodule Myapp.Post do
  use Myapp.Web, :model

  schema "posts" do
    field :title, :string
    field :body, :string

    timestamps()
  end

  def changeset(struct, params \\ %{}) do
    struct
    |> cast(params, [:title, :body])
    |> validate_required([:title, :body])
  end
end
```

CONTROLLER

```
defmodule Myapp.PostController do
  use Myapp.Web, :controller

  def index(conn, _params)                              # GET /tasks
  def new(conn, _params)                                # GET /tasks/new
  def create(conn, %{"post" => post})                   # POST /tasks
  def show(conn, %{"id" => id})                         # GET /tasks/:id
  def edit(conn, %{"id" => id})                         # GET /tasks/:id/edit
  def update(conn, %{"id" => id, "post" => post}) # PUT /tasks/:id
  def delete(conn, %{"id" => id})                       # DELETE /tasks/:id
end
```

TEMPLATE (SHOW)

```
<h2>Show post</h2>

<ul>
  <li>
    <strong>Title:</strong>
    <%= @post.title %>
  </li>
  <li>
    <strong>Body:</strong>
    <%= @post.body %>
  </li>
</ul>

<%= link "Edit", to: post_path(@conn, :edit, @post) %>
<%= link "Back", to: post_path(@conn, :index) %>
```

TEMPLATE (EDIT)

```
<h2>Edit post</h2>

<%= form_for @changeset, post_path(@conn, :update, @post), fn f -> %>
  <%= label f, :title %>
  <%= text_input f, :title %>
  <%= error_tag f, :title %>

  <%= label f, :body %>
  <%= textarea f, :body %>
  <%= error_tag f, :body %>

  <%= submit "Submit" %>
<% end %>

<%= link "Back", to: post_path(@conn, :index) %>
```

AUTHENTIFIZIERUNG

HANDS-ON

WEBSOCKETS

EXKURS

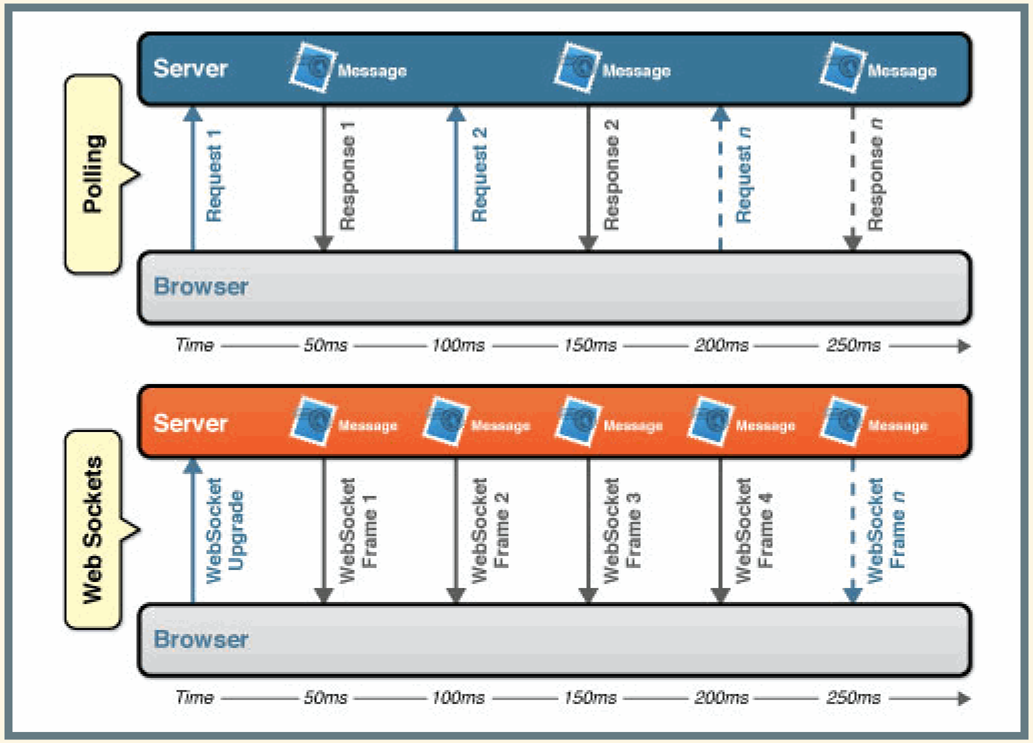
WEBSOCKETS

- Web-Standard
- Permanente Verbindung zum Server
- Ermöglichen Real Time Updates

WEBSOCKETS VS. HTTP

	WebSocket	HTTP
Overhead	2 Bytes	>100 Bytes
Duplex	Vollduplex	Halbduplex
Push	Ja	Nein
Latenz	~50 ms	~150 ms

LATENZ



http://websocket.org/quantum.html

CHAT

HANDS-ON