# Node.js, Adonis.js Vue SSR, Sass

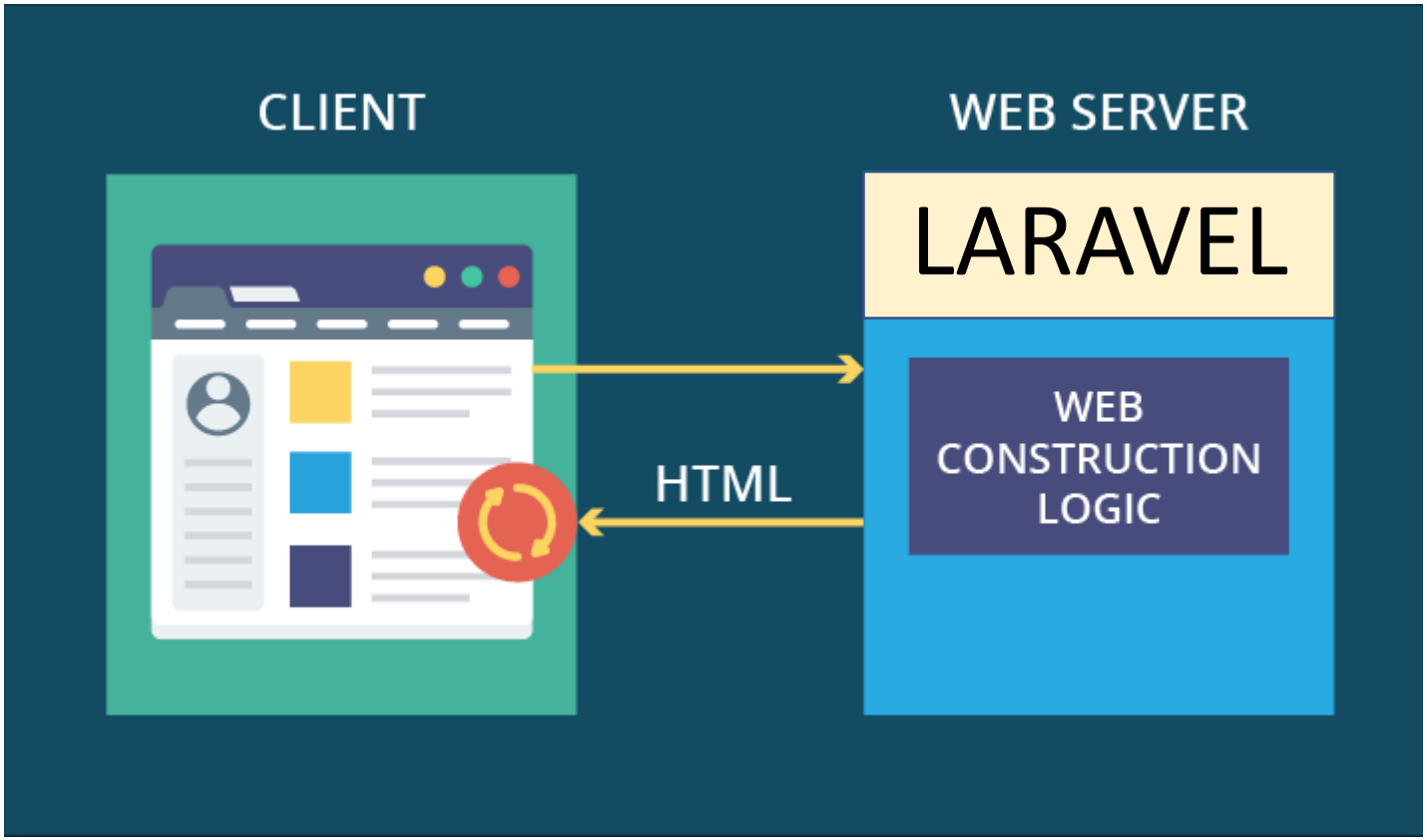
kurz Webové technológie

Eduard Kuric

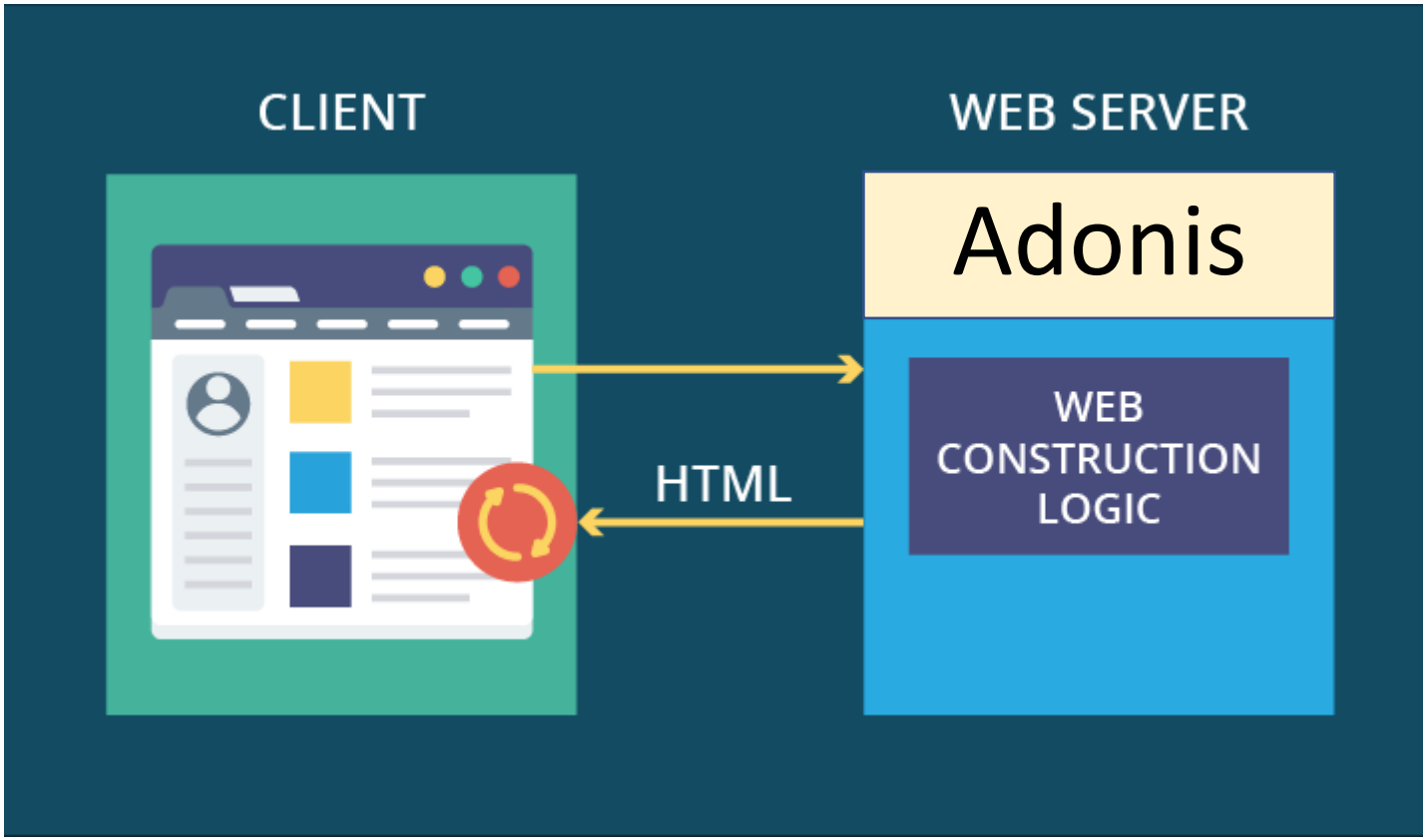
STU FIIT

15. 12. 2020

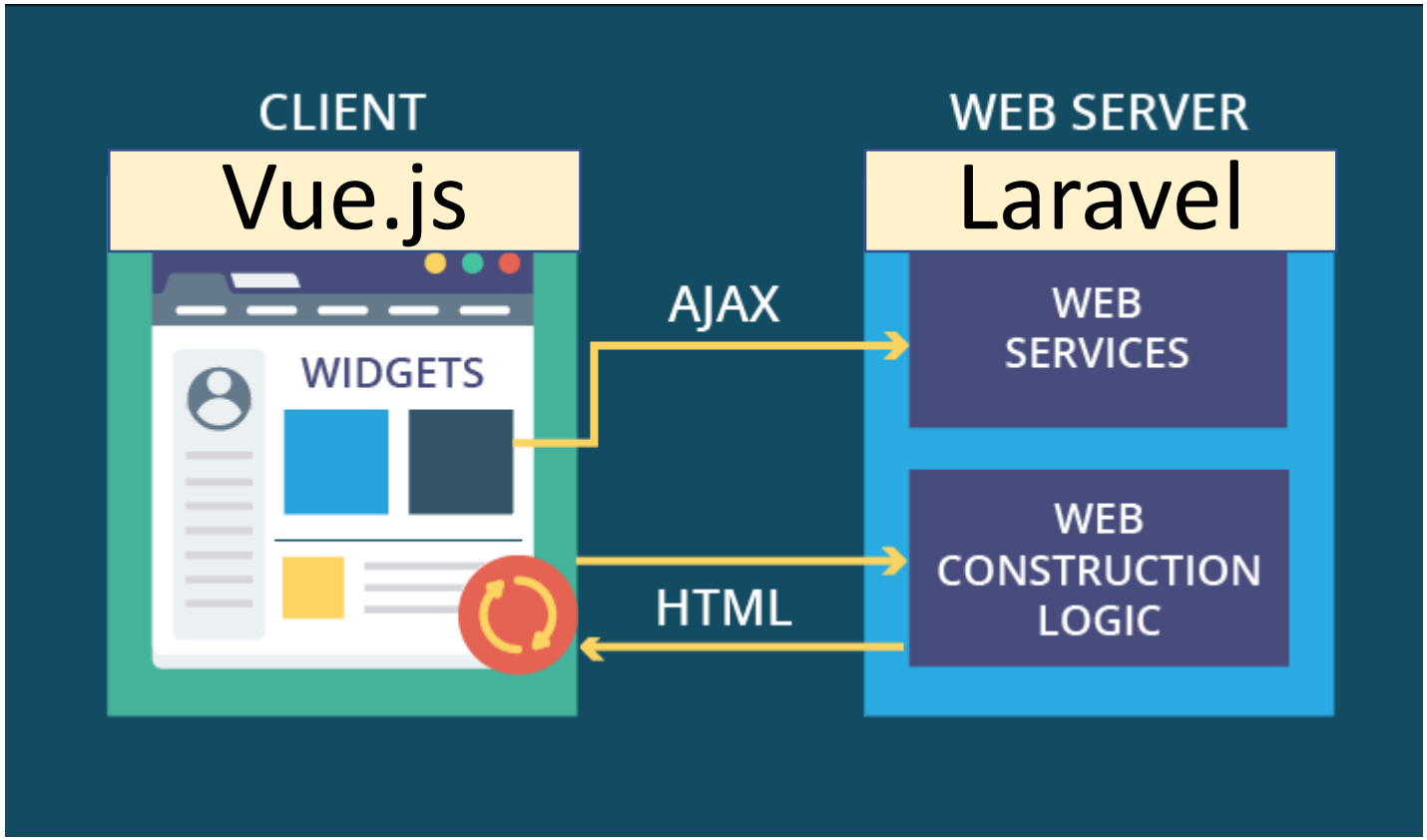
# SSR PHP Laravel



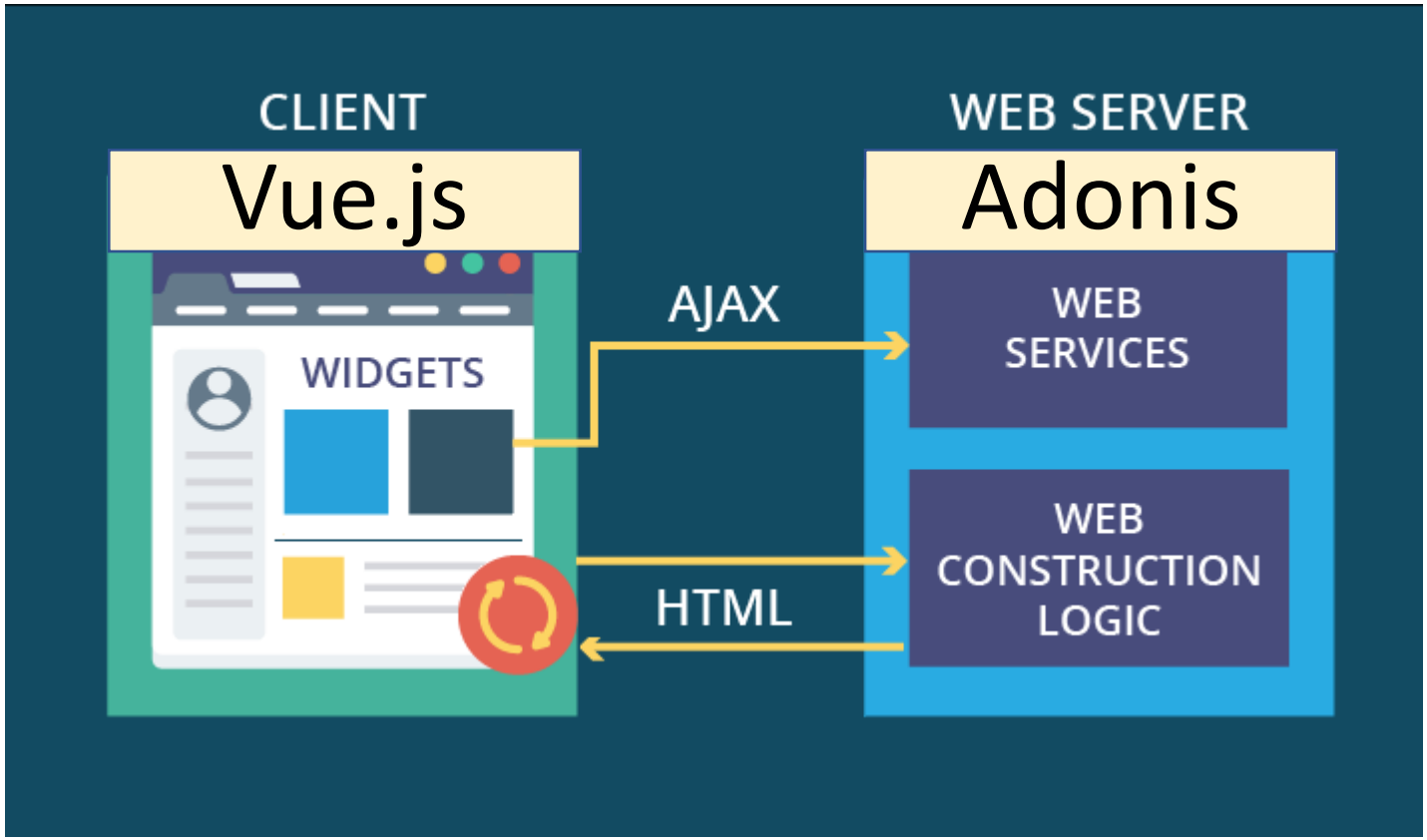
# SSR Node.js (Adonis.js)



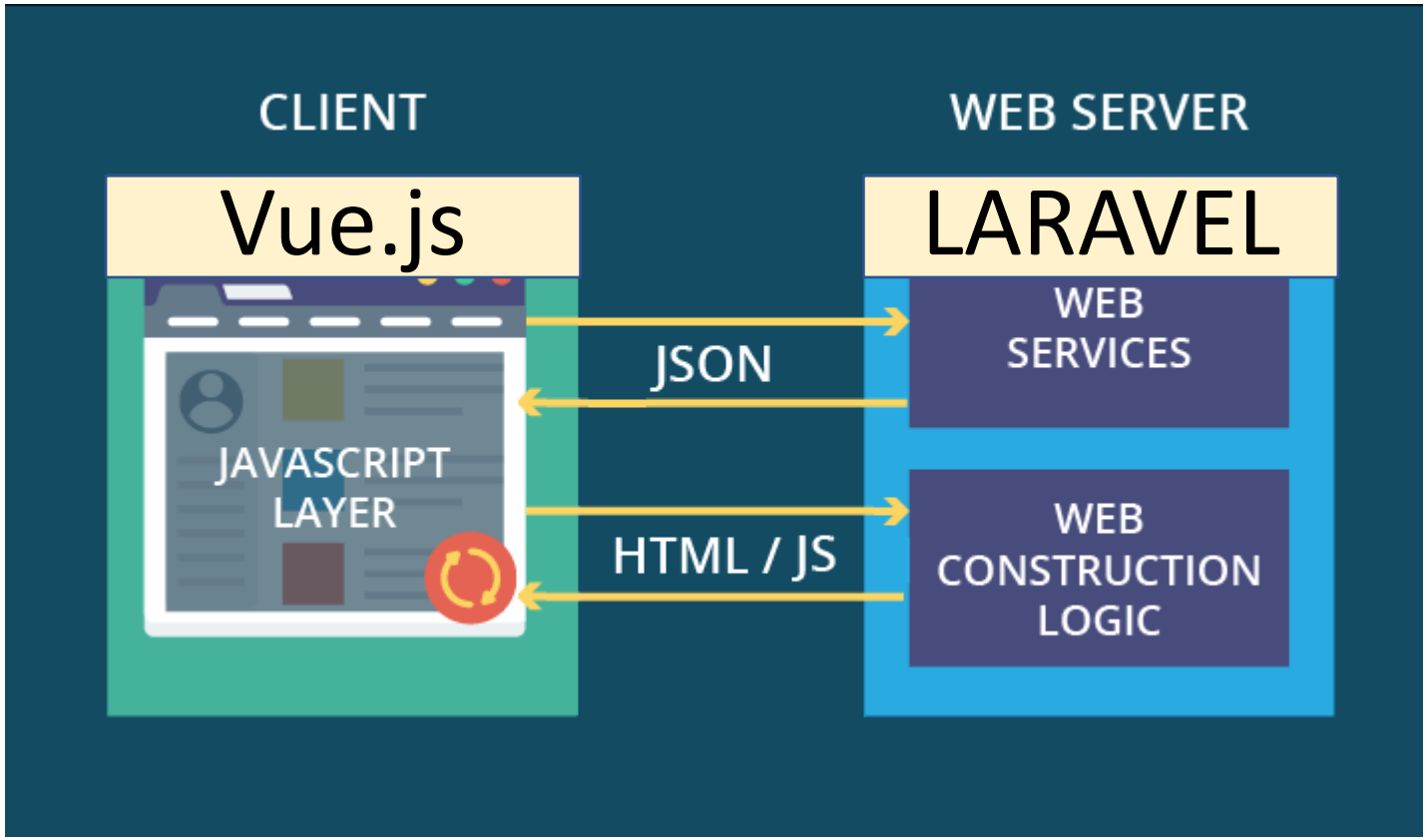
# Vue.js widgets – PHP Laravel



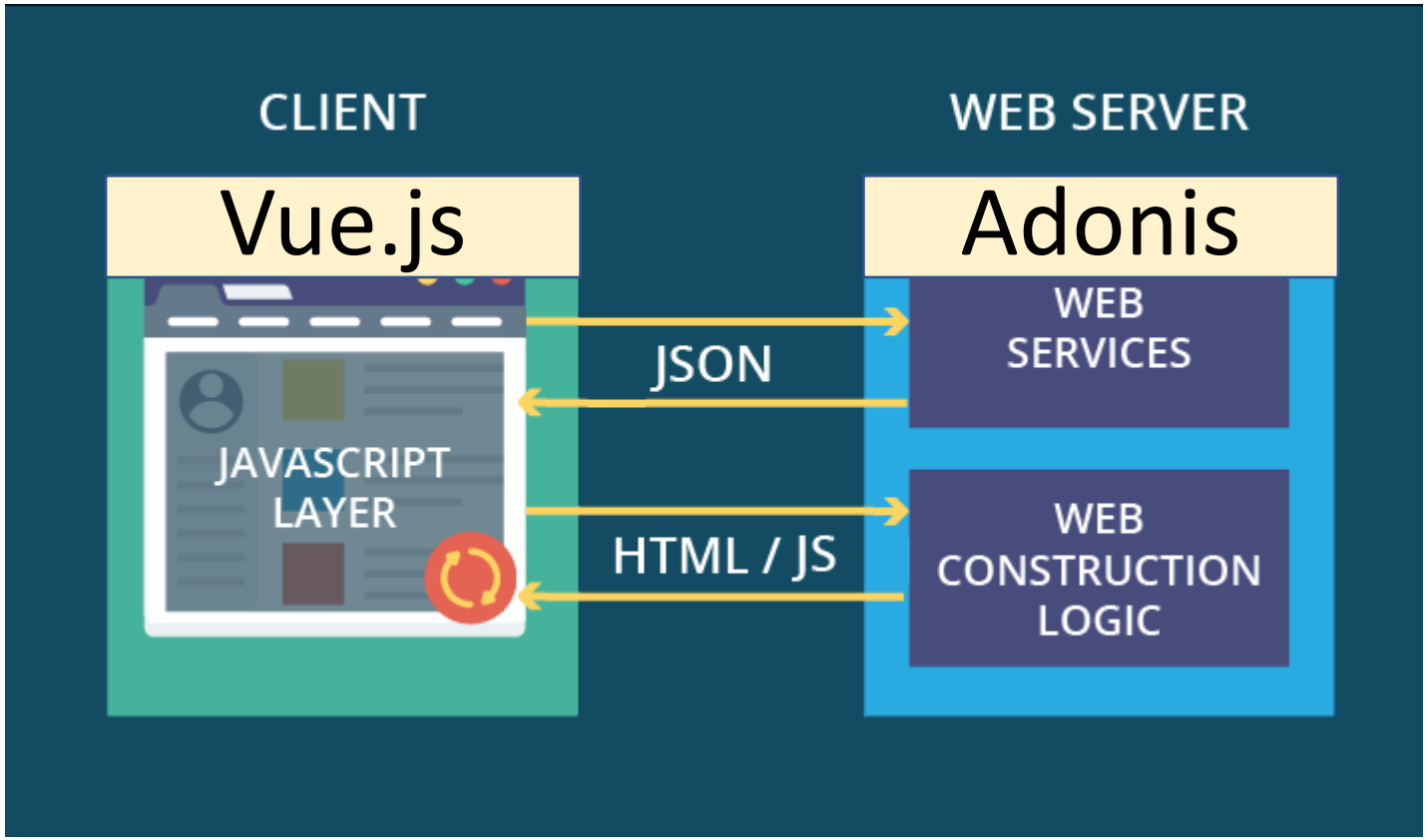
# Vue.js widgets – Adonis.js



# Vue.js SPA – PHP Laravel



# Vue.js SPA – Adonis.js



# Node.js

- prostredie (runtime systém) umožňujúce **vykonávať JavaScript na strane servera**
  - založený na Google V8 engine (ako Chrome)
- otvorený zdrojový kód (open source, MIT licencia)
- asynchrónne, udalosťami riadené API (event-driven)

- zabudované moduly (napr. http, url, events)

# Node.js - `myFirstNodeApp.js`

- HTTP modul
  - umožňuje vytvoriť HTTP server na prenos dát cez HTTP protokol, počúvajúci na určenom porte

```
var http = require('http');
http.createServer(function (req, res) {
  // ak je odpoved HTML obsah, pridame hlavicku
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

- `req` - (objekt) požiadavka od klienta
- `res` - odpoveď zo servera

- **`node myFirstNodeApp.js`**

# Node.js – URL modul

- párovanie URL

```
var url = require('url');
// req.url
var adr =
'http://localhost:8080/products?
        year=2017&month=february';
// true, ak chceme parsovat aj query retazec
var q = url.parse(adr, true);

q.host;      // 'localhost:8080'
q.pathname;        // '/products'
q.search;    // '?year=2017&month=february'

// sparsovany query retazec
// { year: 2017, month: 'february' }
var qdata = q.query;
```

# Node.js – práca so súbormi

- vytváranie, vymazanie, zmena, premenovanie, čítanie zo súborov

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
 fs.readFile('demofile.html', function(err, data){
   res.writeHead(200, {'Content-Type': 'text/html'});
   res.write(data);
   res.end();
 });
}).listen(8080);
```

# Node.js – práca so súbormi /2

```
// ak neexistuje vytvori subor
fs.appendFile('mynewfile.txt', 'Hello content!',
function (err) {});

// ak neexistuje vytvori subor, w - iba na zapis
fs.open('mynewfile2.txt', 'w',
function (err, file) {});

// nahradi existujuci subor/obsah, ak existuje
fs.writeFile('mynewfile3.txt', 'Hello content!',
function (err) {});
```

# Node.js – práca so súbormi /3

```
// vymazanie suboru

fs.unlink('mynewfile2.txt',
function (err) {});


// premenovanie suboru

fs.rename('mynewfile1.txt', 'myrenamedfile.txt',
function (err) {});
```

# Node.js - udalosti

- modul `events`
  - vytváranie, načúvanie a vyvolanie udalostí
  - `EventEmitter object`

```
var events = require('events');
var eventEmitter = new events.EventEmitter();

// vytvorenie obsluhy pre udalost
var myEventHandler = function () {
  console.log('I hear a scream!');
}

// priradenie obsluhy k udalosti
eventEmitter.on('scream', myEventHandler);

// vyvolanie udalosti
eventEmitter.emit('scream');
```

# Express.js

- minimalistický webový rámec nad Node.js poskytujúci základný aparát na zjednodušenie vývoja (MVC) aplikácií

- prináša podporu:
  - smerovanie (routing)
  - middleware
  - šablóny
  - integráciu s DB systémami

# Express.js – ukážka

```
const express = require('express')

const app = express()

app.get('/', (req, res) =>
                res.send('Hello World!'))

app.listen(8080, () =>
                console.log('Listening 8080...'))
```

```
var http = require('http');

http.createServer(function (req, res) {
  // musime napisat logiku pre smerovanie
}).listen(8080);
```

# Vue.js - SSR

- Vue.js je **rámec na vytváranie aplikácií na strane klienta** (v prehliadači)
  - Vue komponenty vytvárajú DOM a manipulujú s ním

```
<html>
<head>…</head>
<body>
…
        <div id="vue-app">
        </div>
…
</body>
</html>
```

```
<script>
var app = new Vue({
    el: '#vue-app',
});
</script>
```

# Vue.js – SSR /2

- **komponenty je možné skompilovať na HTML (HTML reťazce) priamo na serveri**
  - HTML obsah je zostavený na strane servera (Server Side Rendering) a odoslaný klientovi
  - **on-the-fly –** na požiadanie **-** za behu

- **`vue-server-renderer`**

# Vue.js – SSR /2

- `npm install vue` **`vue-server-renderer`** `–save`

- vytvorenie Vue inštancie

```
const app = new Vue({
  template: '<div>Hello World</div>'
})
```

- vytvorenie inštancie **`renderer`**

```
const renderer = require('vue-server-renderer')
                 .createRenderer()
```

# Vue.js – SSR /3

- zostavenia HTML obsahu

```
renderer.renderToString(app).then(html =>
      { console.log(html) }).catch(err =>
      { console.error(err) })
```

# Vue.js – SSR + Express.js príklad

- `node vue-ssr.js`

# Vue.js - prerendering

- za účelom zlepšenia SEO marketingových stránok (/contact, /about) môže postačiť prerendering
- miesto kompilácie komponentov serverom do HTML (on-the-fly), **prerendering vygeneruje statický HTML obsah (pri kompilácii app) pre špecifikované smerovanie (routes)**

- vhodné na stránky neobsahujúce dynamický obsah

- Webpack plugin prerender-spa

# Hydration - hydratácia

- prijatý (statický) HTML obsah (DOM) môžeme na klientovi **hydratovať** na dynamický obsah (reaktívny virtuálny DOM)
  - hydratácia je znovunavrátenie reaktívnej podoby statickému HTML obsahu (prijatému zo serveru)
    - statický DOM -> dynamický DOM

# Hydration /2

- uvažujme, že server vráti

```
<html>
<head>…</head>
<body>
…
<div class="app" id="app" server-rendered="true">
  <div class="labrador">Hello Labrador</labrador>
  <div class="husky"></div>
</div>
…
</body>
</html>
```

# Hydration /3

- kód klienta

```
Vue.component('husky', {
    template: '<div class="husky">Hello husky</div>'
})

var rootComponent = {
    template: '' +
        '<div class="app" id="app">' +
        '    <div class="labrador"></div>' +
        '    <husky></husky>' +
        '</div>'
}

new Vue({
    el: '#app',
    render: h => h(rootComponent)
})
```

# Hydration /3

- Vue nájde **mapovanie medzi DOMom vráteným zo servera a virtuálnym domom vygenerovaným klientskou aplikáciou**
  - tomuto hovoríme hydratácia
  - atribút **`server-rendered`** indikuje Vue, aby sa pokúsil o hydratáciu existujúceho DOMu, namiesto vytvorenia nových DOM uzlov

```
<div class="app" id="app" server-rendered="true">
  <div class="labrador">Hello Labrador</labrador>
  <div class="husky"></div>
</div>

<div class="app" id="app" server-rendered="true">
  <div class="labrador">Hello Labrador</labrador>
  <div class="husky">Hello husky</div>
</div>
```

# Hydration – pozor

- pozor na HTML štruktúru, ktorá môže byť prehliadačom natívne pozmenená
- ak napíšeme v HTML šablóne

```
<table>
    <tr><td>ahoj</td></tr>
</table>
```

prehliadač automaticky pridá `<tbody>`, ALE

virtuálny DOM vytvorený Vue.js neobsahuje `<tbody>`, čo spôsobí nesúlad

- **ak chceme zaistiť správne mapovanie, je potrebné uistiť sa, že v šablónach píšeme platný HTML obsah (štruktúru)**

# Adonis.js

- plnohodnotný MVC rámec
  - veľmi podobný Laravelu, jazyk JavaScript
- podobne ako každý rámec**, Adonis nám umožňuje redukovať množstvo kódu, ktoré potrebujeme napísať** a ušetriť tak čas

- poskytuje rozsiahlu podporu:
  - smerovanie, middleware
  - sedenia (sessions)
  - oprávnenia (policies)
  - šablóny
  - lokalizácia
  - logovanie
  - REST API, websockety
  - ORM
  - služby (odoslanie emailu)
  - nahrávanie súborov
  - testovanie

# Adonis.js – nový projekt

- **npm i -g @adonisjs/cli**

- **adonis new** new-project
- **cd** new-project

- **adonis serve --dev**

# Adonis – štruktúra projektu

- Podobná Laravelu

- `app/`
- `config/`
- `database/`
- `public/`
- `resources/`
- `storage/`
- `start/routes.js`

# Adonis – smerovanie (routing)

- **start/routes.js**

```
// App/Controllers/Http/PostController

// method index()
```

- `Route.get('posts', 'PostController.index')`

- `GET, POST, PUT, PATCH, DELETE`

# Adonis – smerovanie (routing) /2

```
Route.resource('users', 'UserController')
```

- `Route.get('users', 'UserController.index').as('users.index')`
- `Route.post('users', 'UserController.store').as('users.store')`
- `Route.get('users/create', 'UserController.create').as('users.create')`
- `Route.get('users/:id', 'UserController.show').as('users.show')`
- `Route.put('users/:id', 'UserController.update').as('users.update')`
- `Route.patch('users/:id', 'UserController.update')`
- `Route.get('users/:id/edit', 'UserController.edit').as('users.edit')`
- `Route.delete('users/:id', 'UserController.destroy').as('users.destroy')`

# Adonis - controller

```
// HTTP controller
```

- `adonis make:controller User --type http`

```
// websocket controller
```

- `adonis make:controller User --type ws   # WS Controller`

# Adonis – controller /

```
'use strict'

class UserController {
  index ({ request, response }) {
    //
  }
}

module.exports = UserController
```

# Adonis - middleware

- server middleware
  - vykoná sa pred routingom
  - ak požadovaný endpoint nie je registrovaný, aj tak sa vykoná
- global middleware
  - ak je endpoint zaregistorvaný, vykoná sa middleware
  - vykonávajú sa v poradí, v akom sú definované
- named middleware
  - asociovaný s konkrétnym endpointom
  - vykonávajú sa v poradí, v akom sú definované
  - `Route.get('posts', 'PostController.index').middleware(['auth'])`

# Adonis – middleware /2

- adonis make:middleware CountryDetector

```
'use strict'
const geoip = require('geoip-lite')
class CountryDetector {
  async handle ({ request }, next) {
    const ip = request.ip()
    request.country = geoip.lookup(ip).country
    await next()
  }
}
module.exports = CountryDetector
```

# Adonis – service provider

```
const { ServiceProvider } = require('@adonisjs/fold')

class MyProvider extends ServiceProvider {
  register () {
    // register bindings
  }

  boot () {
    // optionally do some initial setup
  }
}

module.exports = MyProvider
```

# Adonis – service provider /2

- Registrácia v `start/app.js`

```
const providers = [
 path.join(__dirname, '..', 'providers', ,MyProvider/Provider')
]
```

# Adonis – šablóny - Edge

- Veľmi podobný Bladu

```
<!DOCTYPE html>
<html lang="en">
<body>
    <div class="container">
      <aside class="sidebar">
        @section('sidebar')
          <p> The default sidebar content </p>
        @endsection
      </aside>
      <div class="content">
        @!section('content')
      </div>
    </div>
  </body>
</html>
```

edge:

```
@layout('master')

@section('content')
  <p> Content area </p>
@endsection
```

blade:

```
@extends('master‘)
...
```

blade:

```
@yield('content')
```

# Adonis – šablóny – Edge /2

- Veľmi podobný Bladu

```
@section('sidebar')
  @super
  <p> Appended to the sidebar </p>
@endsection
```

blade:

```
@section('sidebar')
  @parent
  <p> Appended to the sidebar </p>
@endsection
```

# Adonis ... ďalšie

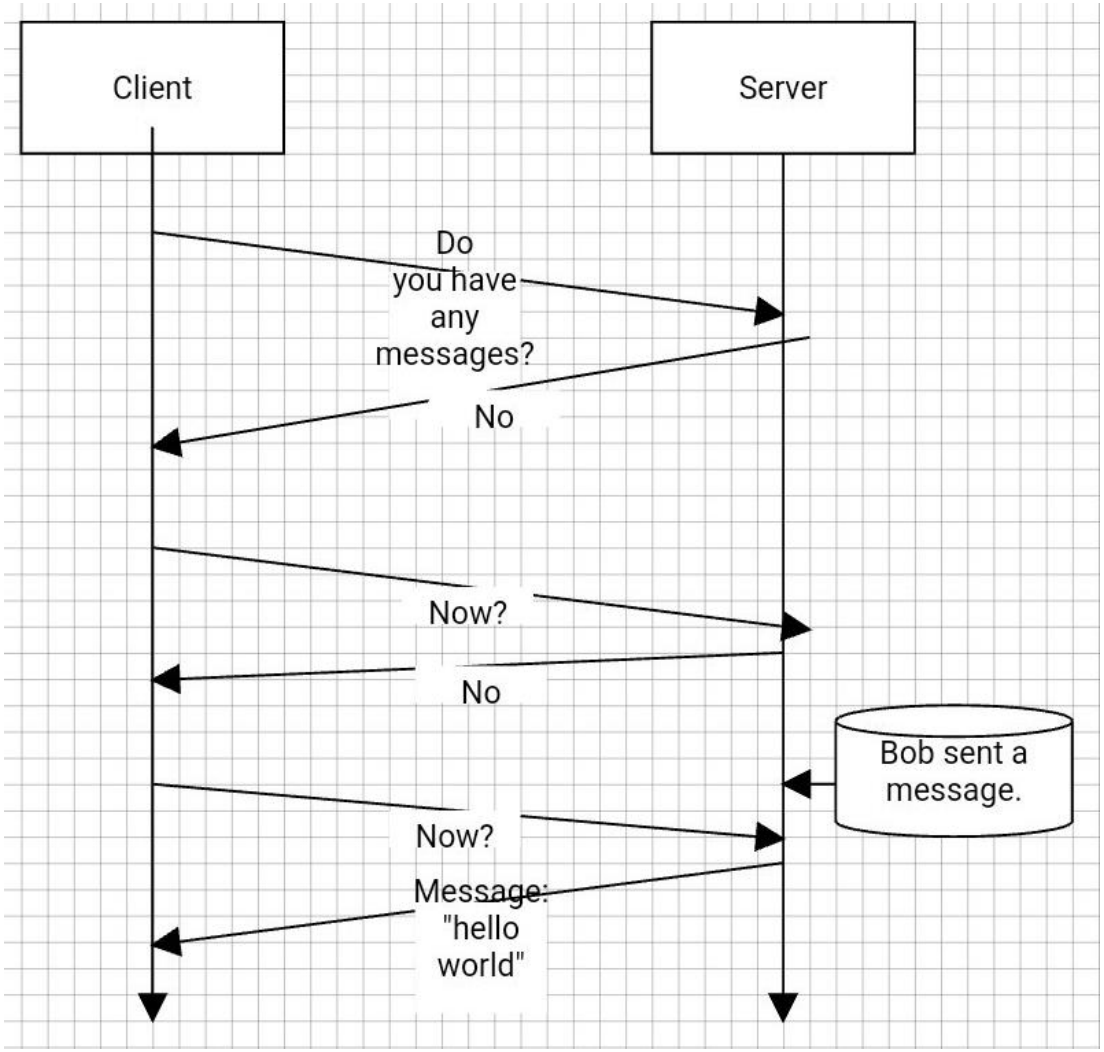
- Sessions
- Validator
- Error handling
- Logger
- Internationalization

# Lokalizácia na klientovi - vue-i18n

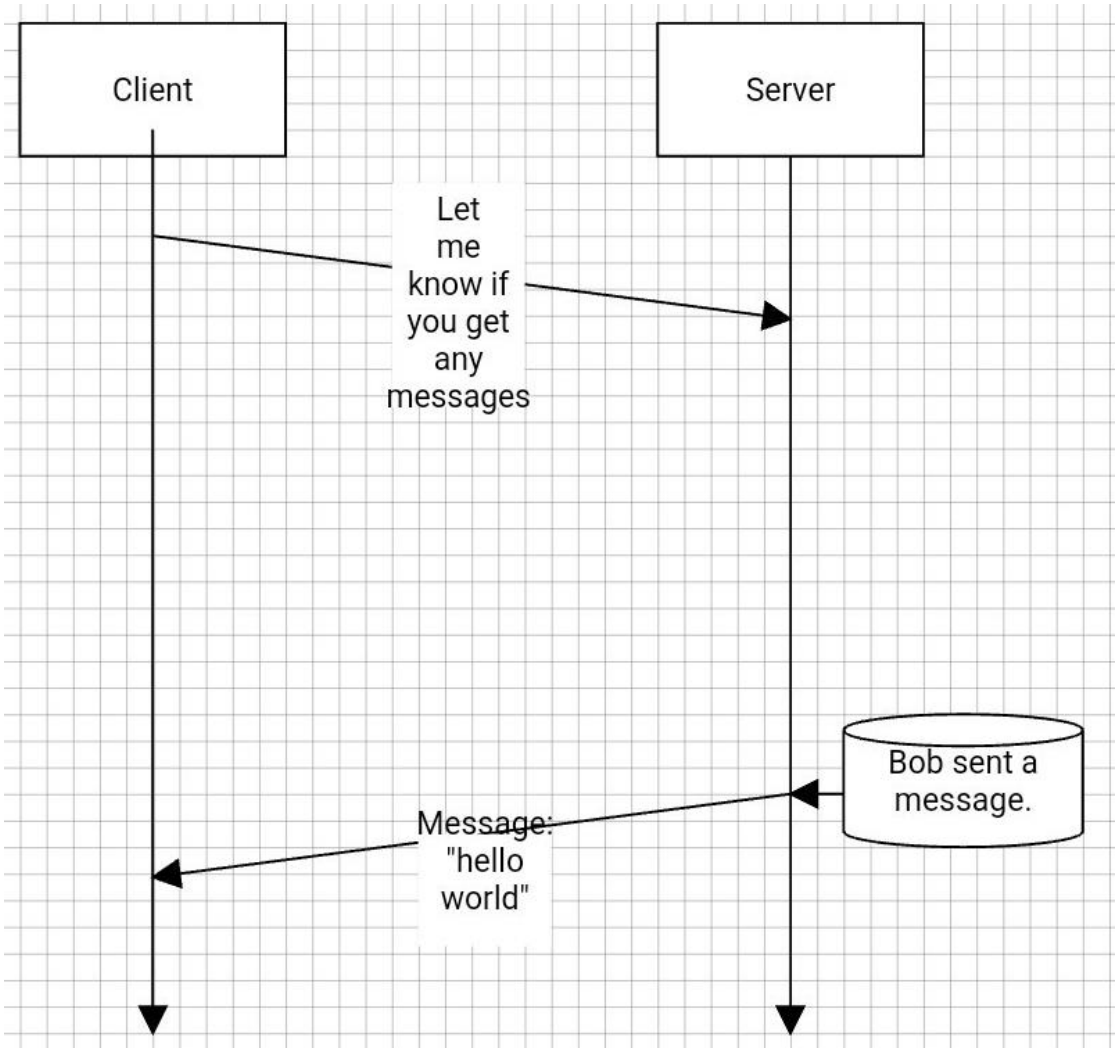
```
const messages = {
  en: {
    message: {
      hello: 'hello world'
    }
  },
  ja: {
    message: {
      hello: 'こんにちは、世界'
    }
  }
}
const i18n = new VueI18n({
  locale: 'ja',   // jazykova mutacia
  messages,       // prekladove retazce
})
new Vue({ i18n }).$mount('#app')
```

```
<div id="app">
 <p>
   {{ $t("message.hello") }}
 </p>
</div>
```

# HTTP vs. websocket



# HTTP vs. websocket /2



# Websocket vs. AJAX

- socket – vytvorené spojenie so serverom pretrváva, (vytvorí sa kanál) – udržiava sa stav
  - AJAX/HTTP požiadavky sú bezstavové
  - server o klientovi nevie
    (iba tak, že si cez cookie prenášajú session id)
- socket - server môže odoslať údaje klientovi (socketu) kedykoľvek
  - v prípade AJAXu - server môže odpovedať klientovi, iba prostredníctvom požiadavky

# Websockety - Adonis

- config/socket.js
  - konfigurácia servera
- start/socket.js
  - boot socket servera a registrovanie kanálov
- start/wsKernel.js
  - registrácia middleware, vykoná sa pri subscribe na kanál

# WS - Jednoduchý chat

- V start/socket.js zadefinujeme kanál

```
const Ws = use('Ws')
Ws.channel('chat', 'ChatController')
```

- Vytvoríme Chat controller

```
adonis make:controller Chat --type=ws
```

# WS – server controller

```
'use strict'

class ChatController {
  constructor ({ socket, request }) {
    this.socket = socket
    this.request = request
  }
}

module.exports = ChatController
```

# WS - klient

```
// use script:
// https://unpkg.com/@adonisjs/websocket-client@1.0.9/dist/Ws.browser.js

<script>
function startChat () {
  ws = adonis.Ws().connect()

  ws.on('open', () => {
    subscribeToChannel()
  })

  ws.on('error', () => {
    …
  })
}
</script>
```

# WS – klient – subscribe na odber ‘message’

```
function subscribeToChannel () {
  const chat = ws.subscribe('chat')

  chat.on('error', () => {
    // …
  })

  chat.on('message', (message) => {
    // append to a list
  })
}
```

# WS – klient – odoslanie správy

```
ws.getSubscription('chat').emit('message', {
      username: username,
      body: message
 })
```

# WS – server - controller

```
class ChatController {
  constructor ({ socket, request }) {
    this.socket = socket
    this.request = request
  }

  // spravu broadcastne vsetkym odberatelom 'message'
  onMessage (message) {
    this.socket.broadcastToAll('message', message)
  }
}
```

# Databáza

- podpora PostgreSQL, MySQL, SQLite3, MariaDB, MSSQL
- query builder, podpora migrácií, seeds
- ORM – Lucid

- config/database.js
  - default sqlite

# Lucid – vytvorenie modelu

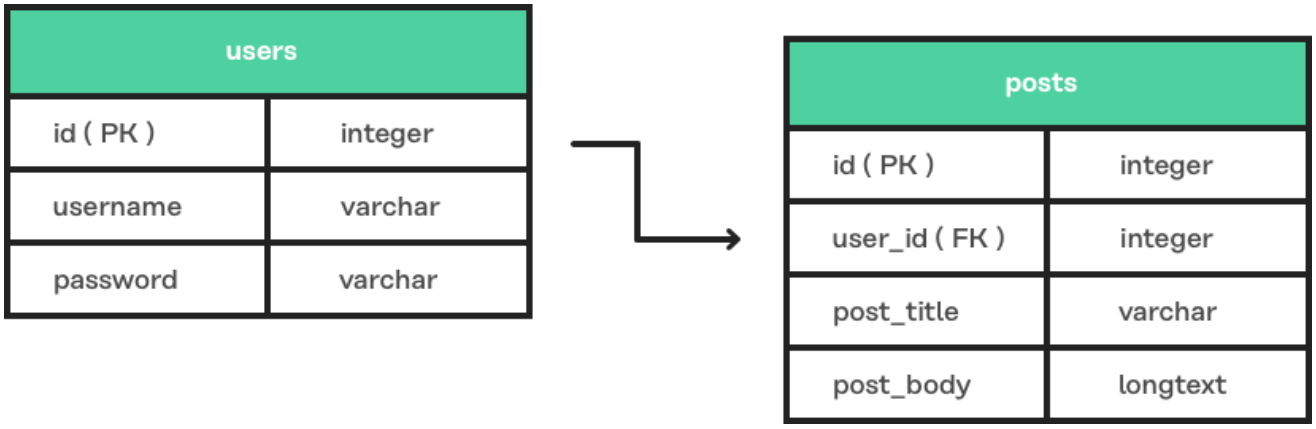
**adonis make:model User –migration**

```
//app/Models/User.js
'use strict'

const Model = use('Model')

class User extends Model {
      ...
}
module.exports = User
```

# Lucid – one-to-many

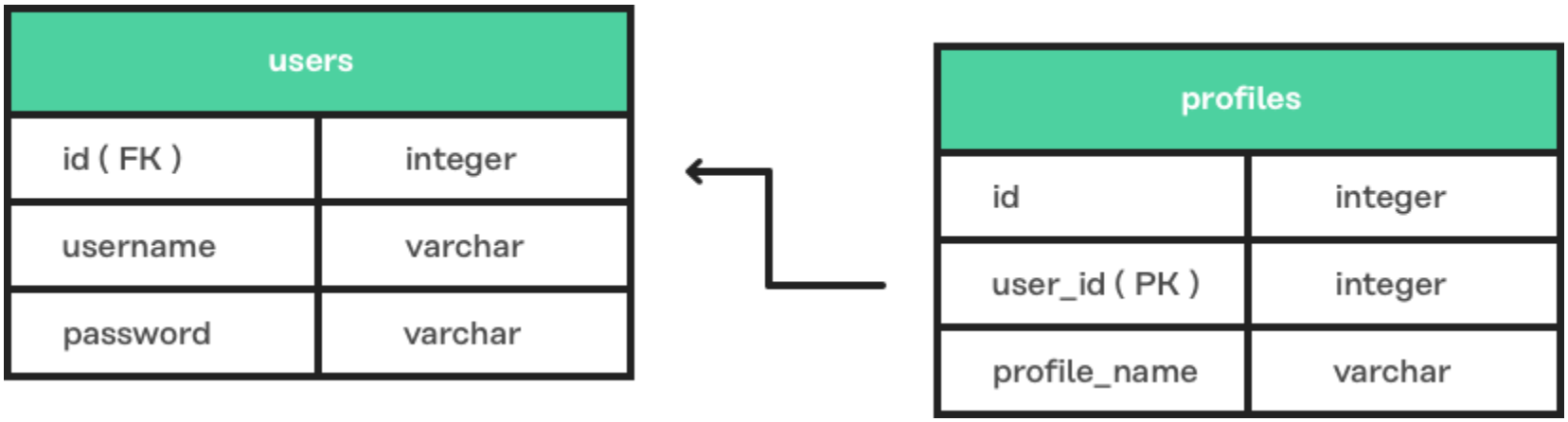



```
const Model = use('Model')

class User extends Model {
  posts () {
    return this.hasMany('App/Models/Post')
  }
}

module.exports = User
```

# Lucid – belongs to




```
const Model = use('Model')

class Profile extends Model {
  user () {
    return this.belongsTo('App/Models/User')
  }
}

module.exports = Profile
```

# Lucid - fetch, create

```
const User = use('App/Models/User')

// fetch all users
await User.all()

// create user
const user = new User()
user.username = 'ekoku'
user.password = '123456'
await user.save()
```

# Sass - Syntactically Awesome Style Sheets

- CSS preprocesor
  - skriptovací jazyk, ktorý rozširuje možnosti písania CSS
  - napísaný kód sa “kompiluje” do CSS
- ďalšie známe preprocesory sú Less a Stylus

- pôvodná - indented **syntax - `.sass`**
- nová syntax (Sass 3) Sassy CSS - `.scss`

# Sass - premenné

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

# Sass - premenné

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

skompilované na

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

# Sass - vnorenie

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
}
```

skompilované na

```
nav ul {
…
}
```

# Sass - partials

- umožňujú modularizovať štýly
- majú špeciálny prefix _
- súbory s daným prefixom sass kompilátor vloží na začiatok hlavného súboru cez direktívu `@import`

- `_reset.scss, base.scss`

```
// base.scss

@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

# Sass - mixins

- umožňujú zoskupiť css deklarácie – podpora znovupoužitia

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

# Sass - mixins

- umožňujú zoskupiť css deklarácie
  – podpora znovupoužitia

skompilované na

```
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

# Sass - dedenie

```
/* using placeholder */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.message {
 @extend %message-shared;
}

.error {
  @extend %message-shared;
  border-color: red;
}
```

```
.message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}
.message {
 @extend .message-shared;
}
.error {
  @extend .message-shared;
  border-color: red;
}
```

# Sass - dedenie

skompilované na

```
.message, .error {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}

.error {
  border-color: red;
}
```

```
.message-shared,
.message, .error {
  border: 1px solid
          #cccccc;
  padding: 10px;
  color: #333;
}
.error {
  border-color: red;
}
```

# Sass - operátory

```
.container { width: 100%; }

article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}
```

# Sass - operátory

skompilované na

```
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 62.5%;
}
```

# Sass – inštalácia/webpack zavádzač

- `npm install -g sass`
- `sass source/stylesheets/index.scss build/stylesheets/index.css`

- Webpack Sass loader