ЗАТВЕРДЖЕНО

1116130.00923-01 12-01-ЛЗ

КЛІЄНТСЬКА ЧАСТИНА МОДУЛЮ «МОНІТОРИНГ УСПІШНОСТІ» для системи «інструменти кафедри»

Текст програми

1116130.00923-01 12-01

Листів 13

2016

АНОТАЦІЯ

Документ 1116130.00923-01 12-01 «Клієнтської частина модулю «Моніторинг успішності» для системи «Інструменти кафедри». Текст програми» входить до складу програмної документації до дипломного проекту.

У даному документі описана структура та текст веб клієнту. Веб клієнт написаний на мовах JavaScript, HTML. Об’єм пам’яті, що займає програма комплексу та конфігураційні файли, складає 30 Мб. Конфігурація комп’ютера стандартна, на ньому повинно бути встановлено середовище розробки WebStorm Community Edition.

ЗМІСТ

[1 Структура програми 4](#_Toc447145138)

[1.1 Структура компоненту auth 4](#_Toc447145139)

[1.2 Структура компоненту groups 4](#_Toc447145140)

[1.3 Структура компоненту teachers 4](#_Toc447145141)

[2 Текст програм 6](#_Toc447145142)

[2.1 Текст файлу authController.coffee 6](#_Toc447145143)

[2.2 Текст файлу loginView.html 6](#_Toc447145144)

[2.3 Текст файлу registerView.html 6](#_Toc447145145)

[2.4 Текст файлу groupController.coffee 7](#_Toc447145146)

[2.5 Текст файлу groupsController.coffee 8](#_Toc447145147)

[2.6 Текст файлу groupsFactory.coffee 8](#_Toc447145148)

[2.7 Текст файлу groupsView.html 9](#_Toc447145149)

[2.8 Текст файлу groupView.html 9](#_Toc447145150)

[2.9 Текст файлу teacherController.coffee 10](#_Toc447145151)

[2.10 Текст файлу teachersController.coffee 11](#_Toc447145152)

[2.11 Текст файлу teachersFactory.coffee 11](#_Toc447145153)

[2.12 Текст файлу teachersView.html 12](#_Toc447145154)

[2.13 Текст файлу teacherView.html 12](#_Toc447145155)

# 1 СТРУКТУРА ПРОГРАМИ

Веб клієнт написаний на мові програмування JavaScript в середовищі програмування WebStorm Community Edition.

Веб клієнт складається з наступних компонентів:

* auth (компонент, що включає логіку та розмітку сторінок для авторизації та реєстрації);
* groups (компонент, що включає логіку та розмітку сторінок для перегляду, створення та редагування груп);
* teachers (компонент, що включає логіку та розмітку сторінок для перегляду, створення та редагування викладачів).

## 1.1 Структура компоненту auth

Файл authController.coffee містить контролер authController – контролер, що включає функції авторизації та реєстрації.

Файл loginView.html містить розмітку сторінки авторизації.

Файл registerView.html містить розмітку сторінки реєстрації.

## 1.2 Структура компоненту groups

Файл groupController.coffee містить контролер groupController – контролер, що включає функції перегляду, оновлення, та створення групи.

Файл groupsController.coffee містить контролер groupsController – контролер, що включає функції перегляду та видалення груп.

Файл groupsFactory.coffee містить сервіс groupsFactory – сервіс, що реалізовує отримання групи по її номеру та отримання усіх груп з серверу, оновлення та створення груп на сервері.

Файл groupsView.html містить розмітку сторінки груп.

Файл groupView.html містить розмітку сторінки групи.

## 1.3 Структура компоненту teachers

Файл teacherController.coffee містить контролер teacherController – контролер, що включає функції перегляду, оновлення, та створення викладача.

Файл teachersController.coffee містить контролер teachersController – контролер, що включає функції перегляду та видалення викладачів.

Файл teacherFactory.coffee містить сервіс teacherFactory – сервіс, що реалізовує отримання викладача по його номеру та отримання усіх викладачів з серверу, оновлення та створення викладача на сервері.

Файл teachersView.html містить розмітку сторінки викладачів.

Файл teacherView.html містить розмітку сторінки викладача.

# 2 ТЕКСТ ПРОГРАМ

## 2.1 Текст файлу authController.coffee

App.angular.controller(

'AuthController'

[

'$scope'

'$state'

'Auth'

($scope, $state, Auth, Notification) ->

$scope.login = () ->

Auth.login($scope.user).then(

() ->

$state.go('navigable.home')

)

$scope.register = () ->

Auth.register($scope.user).then(

()->

$state.go('navigable.home')

)

]

)

## 2.2 Текст файлу loginView.html

<div class="container-fluid">

<div class="page-header">

<h1>Log In</h1>

</div>

<form ng-submit="login()">

<div class="input-group">

<input type="email" name="email" class="form-control" placeholder="Email" ng-model="user.email">

</div>

<div class="input-group">

<input type="password" name="password" class="form-control" placeholder="Password" ng-model="user.password">

</div>

<div class="btn-group" role="group">

<button type="submit" class="btn btn-default">Log In</button>

</div>

</form>

</div>

## 2.3 Текст файлу registerView.html

<div class="page-header">

<h1>Register</h1>

</div>

<form ng-submit="register()">

<div class="input-group">

<input type="email" name="email" class="form-control" placeholder="Email" ng-model="user.email">

</div>

<div class="input-group">

<input type="password" name="password" class="form-control" placeholder="Password" ng-model="user.password">

</div>

<input type="submit" class="btn btn-default" value="Register">

</form>

## 2.4 Текст файлу groupController.coffee

'use strict';

groups = (groupFactory, $state, $http)->

vm = @

vm.group = {}

vm.student = {}

vm.students = []

editing = false

if !isNaN($state.params.id)

editing = true

groupFactory.getItem($state.params.id).then((e)->

vm.group = e.data

)

vm.save = (form)=>

if !editing

groupFactory.insert(form).then((e)->

if e.data

$state.go('navigable.group', {id: e.data.id})

else

$state.go('navigable.groups')

)

else

groupFactory.update(vm.group.id, form)

vm.addStudent = (student)=>

groupFactory.new\_student(vm.group.id, student.id).then(()=>

vm.group.students.push(student)

vm.student = {}

)

vm.refreshStudents = (text)=>

params = {text: text}

return $http.get('/students/search', {params: params})

.then((response)=>

vm.students = response.data

);

return vm

groups.$inject = ['groupsFactory', '$state', '$http']

App.angular.controller('groupController', groups)

## 2.5 Текст файлу groupsController.coffee

'use strict';

groups = (groupsFactory)->

vm = @

vm.groups = []

@getItems = ()->

groupsFactory

.getItems()

.success((groups)->

vm.groups = groups

return vm.groups

)

@getItems()

return vm

groups.$inject = ['groupsFactory']

App.angular.controller('groupsController', groups)

## 2.6 Текст файлу groupsFactory.coffee

App.angular.factory(

'groupsFactory'

[

'$http'

($http) ->

@f = {}

@page = 1

@base\_url = 'groups'

@f.getItems = ()=>

$http.get(@base\_url)

@f.getItem = (id)=>

$http.get("#{@base\_url}/#{id}")

@f.insert = (model)=>

$http.post(@base\_url, model)

@f.update = (id, model)=>

param = {

id: id,

group: model

}

$http.put("#{@base\_url}/#{id}",param)

@f.new\_student = (id, studentId) =>

$http.post("#{@base\_url}/#{id}/add\_student/#{studentId}")

@f

]

)

## 2.7 Текст файлу groupsView.html

<div class="row">

<a href="#/groups/new" class="btn btn-info">New Group</a>

</div>

<br>

<div class="row">

<table class="table table-bordered">

<thead>

<tr>

<th>Group Name</th>

<th>Some Actions</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="group in vm.groups">

<td><a href="#/groups/{{::group.id}}">{{::group.name}}</a></td>

<td>Some Actions</td>

</tr>

</tbody>

</table>

</div>

## 2.8 Текст файлу groupView.html

<div>

<h4>New group form</h4>

<form name="groupForm" ng-submit="vm.save(vm.group)" novalidate ng-cloak>

<div class="form-group">

<label for="first\_name">Name</label>

<input type="text" name="first\_name" class="form-control" ng-model="vm.group.name" required>

</div>

<div class="form-group">

<label for="year">Year</label>

<input type="date" name="year" class="form-control" ng-model="vm.group.year" >

</div>

<hr>

<h4>Students in group</h4>

<table class="table table-condensed">

<thead>

<tr>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<tr >

<td>

<ui-select ng-model="vm.student" theme="bootstrap" reset-search-input="false">

<ui-select-match placeholder="Select a student">{{$select.selected.full\_name}}</ui-select-match>

<ui-select-choices repeat="student in vm.students "

refresh="vm.refreshStudents($select.search)"

refresh-delay="0">

<div ng-bind-html="student.full\_name | highlight: $select.search"></div>

</ui-select-choices>

</ui-select>

</td>

<td>

<button type="button" class="btn btd-default" ng-click="vm.addStudent(vm.student)">Add</button>

</td>

</tr>

<tr ng-repeat="student in vm.group.students">

<td>{{::student.first\_name}}</td>

<td></td>

</tr>

</tbody>

</table>

<button type="submit" class="btn btn-primary" ng-disabled="groupForm.$invalid" ng-bind="vm.group.id ? 'Save': 'Create'"></button>

<button type="button" class="btn btn-default" back-button>Back</button>

</form>

</div>

## 2.9 Текст файлу teacherController.coffee

'use strict';

teacher = (teacherFactory, $state)->

vm = @

vm.teacher = {}

editing = false

if !isNaN($state.params.id)

editing = true

teacherFactory.getItem($state.params.id).then((e)->

vm.teacher = e.data

)

vm.save = (form)=>

if !editing

teacherFactory.insert(form).then((e)->

if e.data

$state.go('navigable.teacher', {id: e.data.id})

else

$state.go('navigable.teachers')

)

else

teacherFactory.updateStudent(vm.teacher.id, form)

return vm

teacher.$inject = ['teachersFactory', '$state']

App.angular.controller('teacherController', teacher)

## 2.10 Текст файлу teachersController.coffee

'use strict';

teachers = (teachersFactory)->

vm = @

vm.teachers = []

@getItems = ()->

teachersFactory

.getItems()

.success((teachers)->

vm.teachers = teachers

return vm.teachers

)

@getItems()

return vm

teachers.$inject = ['teachersFactory']

App.angular.controller('teachersController', teachers)

## 2.11 Текст файлу teachersFactory.coffee

App.angular.factory(

'teachersFactory'

[

'$http'

($http) ->

@f = {}

@page = 1

@base\_url = 'teachers'

@f.getItems = ()=>

$http.get(@base\_url)

@f.getItem = (id)=>

$http.get("#{@base\_url}/#{id}")

@f.insert = (model)=>

$http.post(@base\_url, model)

@f.update = (id, model)=>

$http.put("#{@base\_url}/#{id}", model)

@f

]

)

## 2.12 Текст файлу teachersView.html

<div class="row">

<a href="#/teachers/new" class="btn btn-info">New Teacher</a>

</div>

<br>

<div class="row">

<table class="table table-bordered">

<thead>

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Some Actions</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="teacher in vm.teachers">

<td><a href="#/teachers/{{::teacher.id}}">{{::teacher.first\_name}}</a></td>

<td><a href="#/teachers/{{::teacher.id}}">{{::teacher.last\_name}}</a></td>

<td>Some Actions</td>

</tr>

</tbody>

</table>

</div>

## 2.13 Текст файлу teacherView.html

<div>

<h3>New student form</h3>

<form name="teacherForm" ng-submit="vm.save(vm.teacher)" novalidate ng-cloak>

<div class="form-group">

<label for="first\_name">First Name</label>

<input type="text" name="first\_name" class="form-control" ng-model="vm.teacher.first\_name" required>

</div>

<div class="form-group">

<label for="last\_name">Last Name</label>

<input type="text" name="last\_name" class="form-control" ng-model="vm.teacher.last\_name" required>

</div>

<div class="form-group">

<label for="middle\_name">Middle Name</label>

<input type="text" name="middle\_name" class="form-control" ng-model="vm.teacher.middle\_name" required>

</div>

<div class="form-group">

<label for="date\_of\_birth">Date of Birth</label>

<input type="date" name="date\_of\_birth" class="form-control" ng-model="vm.teacher.date\_of\_birth">

</div>

<button type="submit" class="btn btn-primary" ng-disabled="teacherForm.$invalid" ng-bind="vm.teacher.id ? 'Save': 'Create'"></button>

<button type="button" class="btn btn-default" back-button>Back</button>

</form>

</div>