**4** РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ

Данный программный продукт состоит из нескольких частей, написанных на разных языках. Детали реализации основных алгоритмов описаны ниже.

**4.1** Алгоритм рассылки приглашений группе пользователей

В web-сервисе доступна функция приглашения списка пользователей из файла в систему для прохождения определенного курса. Она позволяет предоставить доступ к курсу сразу целой группе пользователей, чьи емэйлы указаны в файле.

Для приглашения группы пользователей необходимо загрузить в систему CSV файл со списком емэйлов пользователей.

Функция self.import(file, organization), реализующая данный алгоритм представлена ниже. Она расположена в файле user.rb и является публичным методом модели User.

def self.import(file, organization)

CSV.foreach(file.path) do |email|

if EmailValidator.valid?(email.first)

user = User.find\_by\_email(email.first)

if user

UserMailer.invitation\_instractions(user.email, organization).deliver\_later

if organization.users.exclude?(user)

organization.users\_organizations.create(user\_id: User.find\_by\_email(email.first).id, state: :invited)

end

else

User.invite!(email: email.first, users\_organizations: [ UsersOrganization.new(organization\_id: organization.id, state: :invited)])

end

end

end

end

Функция принимает два аргумента:

* file – имя файла со списком емэйлов пользователей;
* organization – организация, в которую необходимо пригласить пользователей.

Далее в цикле осуществляется перебор всех емэйлов в файле.

CSV.foreach(file.path) do |email|

Проверяется валидность емэйла.

if EmailValidator.valid?(email.first)

Если адрес электронной почты валиден, осуществляется поиск пользователя с таким емэйлом в системе.

user = User.find\_by\_email(email.first)

Если пользователь с таким емэйлом существует в системе, и он еще не является членом организации, то ему почту отправляется письмо с приглашением вступить в организацию.

UserMailer.invitation\_instractions(user.email,organization).deliver\_later

В противном случае на адрес электронной почты высылается приглашение для регистрации в web-сервисе.

User.invite!(email: email.first, users\_organizations: [ UsersOrganization.new(organization\_id: organization.id, state: :invited)])

**4.2** Алгоритм расчета прогресса прохождения курса

Для того, чтобы определить успешность прохождения пользователем курса используется функция def self.percent(course\_id, user), которая является публичным методом модели Course.

def self.percent(course\_id, user)

pages\_id = Page.where(course\_id: course\_id, page\_type: "Question")

question\_count = pages\_id.count

correct\_answers\_count = 0.0

if question\_count == 0

100

else

pages\_id.each do |page|

system\_answer = Answer.where(page\_id: page.id)

input\_user\_answer = InputUserAnswer.where(page\_id: page.id, user\_id: user.id)

case system\_answer.first.answer\_type

when "Input"

correct\_answers\_count += check\_input(input\_user\_answer)

when "Radio"

correct\_answers\_count += check\_radio(input\_user\_answer)

else

correct\_answers\_count += check\_checkbox(input\_user\_answer, page.id)

end

end

correct\_answers\_count / question\_count \* 100

end

end

Функция принимает два аргумента:

* course\_id – уникальный идентификатор курса;
* user – объект модели класса User.

Далее внутри функции осуществляется поиск все материалов, относящихся к данному курсы, в которых содержатся вопросы.

pages\_id = Page.where(course\_id: course\_id, page\_type: "Question")

И сохраняется их количество в переменную question\_count.

question\_count = pages\_id.count

Значение количества правильных ответов устанавливается в ноль.

correct\_answers\_count = 0.0

Далее в цикле осуществляется перебор всех вопросов курса.

pages\_id.each do |page|

Для каждого вопроса загружается и сохраняется в переменную system\_answer правильный ответ, хранящийся в системе, который представляет из себя объект модели Answer.

system\_answer = Answer.where(page\_id: page.id)

Далее загружается и сохраняется в переменную input\_user\_answer введенный пользователем ответ.

input\_user\_answer = InputUserAnswer.where(page\_id: page.id, user\_id: user.id)

После чего проверяется тип вопроса, и в зависимости от него вызывается соответствующая функция для проверки правильности введенного пользователем ответа, каждая из которых возвращает 1, если пользователь ввел правильный ответ и 0 в противном случае.

case system\_answer.first.answer\_type

Если тип вопроса «Input» – вызывается функция check\_input(input\_user\_answer).

def self.check\_input(answer)

(answer.first && answer.first.answer\_id.nil?) ? 0 : 1

end

Если тип вопроса «Radio» – вызывается функция check\_input(input\_user\_answer).

def self.check\_radio(answer)

result = Answer.find(answer.first.answer\_id)

if result.is\_right

1

else

0

end

end

В противном случае – вызывается функция check\_input(input\_user\_answer)

def self.check\_checkbox(answer, page\_id)

tmp\_system\_answer = Answer.where(page\_id: page\_id, is\_right: true)

if tmp\_system\_answer.count != answer.count

0

else

counter = 0

tmp\_system\_answer.each\_index do |index|

if tmp\_system\_answer[index].id != answer[index].answer\_id

counter += 1

end

end

if counter == 0

1

else

0

end

end

end

После чего счетчик количества правильных ответов пользователя увеличивается на результат выполнения одной из перечисленных выше функций.

correct\_answers\_count += check\_checkbox(input\_user\_answer, page.id)

После завершения цикла производится преобразование прогресса выполнения курса к процентному виду.

correct\_answers\_count / question\_count \* 100

**4.3** Алгоритм поиска получателя сообщения

Для того, чтобы найти адресата сообщения используется функция choose\_recipients(), написанная на языке программирования JavaScript. Она позволяет динамически подгружать список потенциальных адресатов сообщения на основе введенного запроса пользователем в строку поиска.

function choise\_recipients(){

$('#recipient\_email').change(function(){

$.ajax({

url: '/conversations/get\_recipients',

datatype: 'json',

method: 'GET'

}).success(function(data){

$('.chosen-select').empty();

var content = '';

data.forEach(function(item, i, arr) {

content = '<option>'+item["email"]+'</option>';

$('.chosen-select').append(content);

});

$('.chosen-select').trigger("chosen:updated");

});

});

$('#recipient\_last\_name').change(function(){

$.ajax({

url: '/conversations/get\_recipients',

datatype: 'json',

method: 'GET'

}).success(function(data){

$('.chosen-select').empty();

var content = '';

data.forEach(function(item, i, arr) {

content = '<option>'+item["last\_name"]+'</option>';

$('.chosen-select').append(content);

});

$('.chosen-select').trigger("chosen:updated");

});

});

};

$(document).ready(choise\_recipients)

$(document).on('page:load', choise\_recipients)

Рассмотрим более подробно реализацию данной функции.

Функция начинает работать сразу после загрузки страницы. При изменении строки ввода адресата сообщения на сервер отправляется AJAX-запрос со следующими параметрами:

* url – адрес метода контроллера, на который отправляется запрос;
* datatype – формат данных запроса;
* method – тип запроса.

Функция отправляет запрос к методу get\_recipients контроллера ConversationsControllet и ожидает ответа от него в формате JSON:

$.ajax({

url: '/conversations/get\_recipients',

datatype: 'json',

method: 'GET'

})

На стороне сервера происходит поиск пользователей и отправка их на сторону клиента в формате JSON:

def get\_recipients

@recipients = User.all

render json: @recipients

end

После этого, при успешном завершении запроса, на стороне клиента очищается текущий список потенциальных адресатов сообщения.

$('.chosen-select').empty();

var content = '';

Далее в цикле производится обход всех пользователей и добавляются их емэйлы в список возможных адресатов сообщения:

.success(function(data){

var content = '';

data.forEach(function(item, i, arr) {

content = '<option>'+item["email"]+'</option>';

$('.chosen-select').append(content); });

В случае, если пользователь выбрал опцию поиск по фамилии, то производится добавление фамилий найденных пользователей в список потенциальных адресатов cообщения:

.success(function(data){

$('.chosen-select').empty();

var content = '';

data.forEach(function(item, i, arr) {

content =

'<option>'+item["last\_name"]+'</option>';

$('.chosen-select').append(content);

});

$('.chosen-select').trigger("chosen:updated");

};