|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика.**

**Отчет**

|  |  |
| --- | --- |
| **по зачетной работе** |  |

Название: ToDo — удобный и веселый планировщик

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35Б |  |  | Л.Э. Барсегян |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

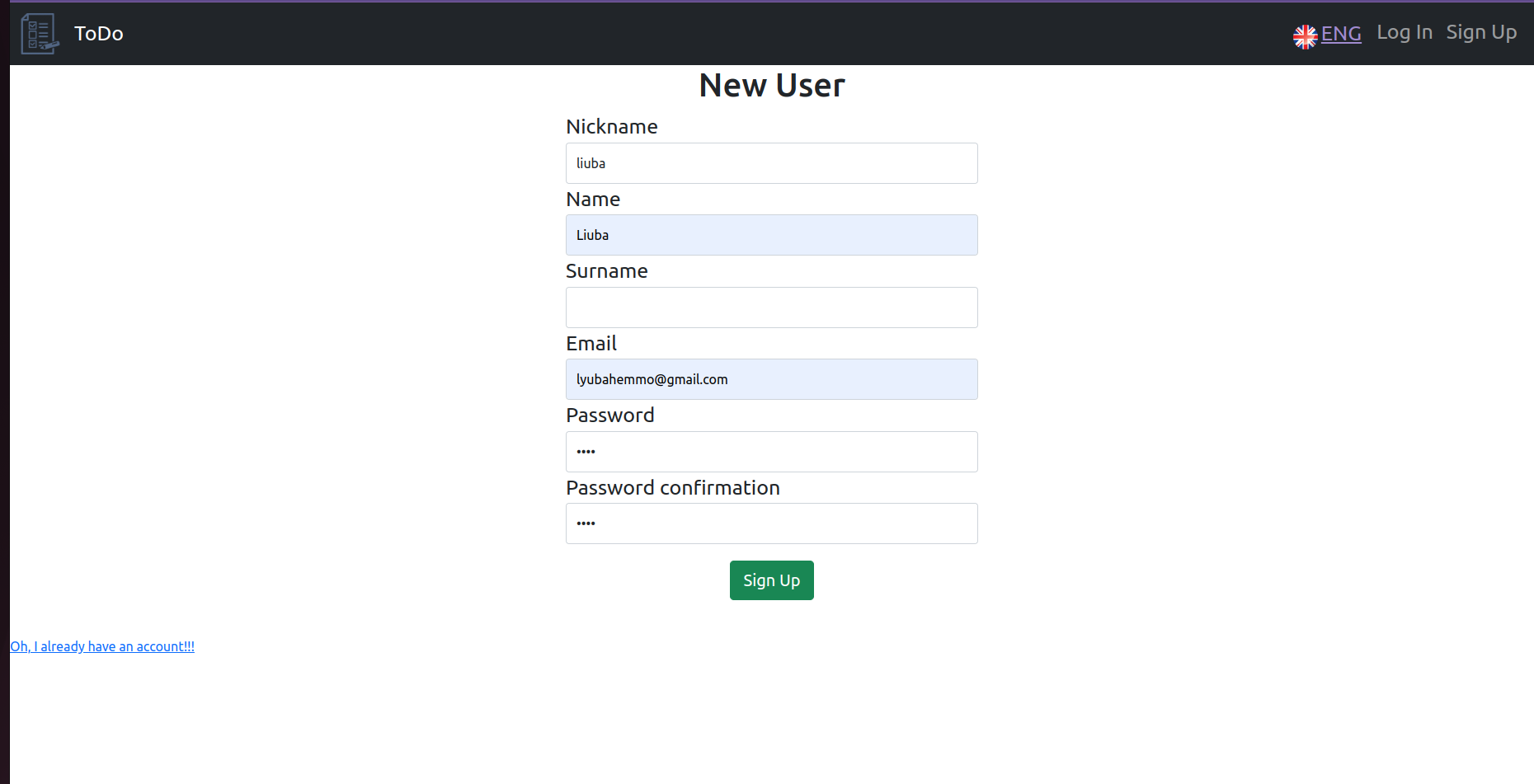
Москва, 2022

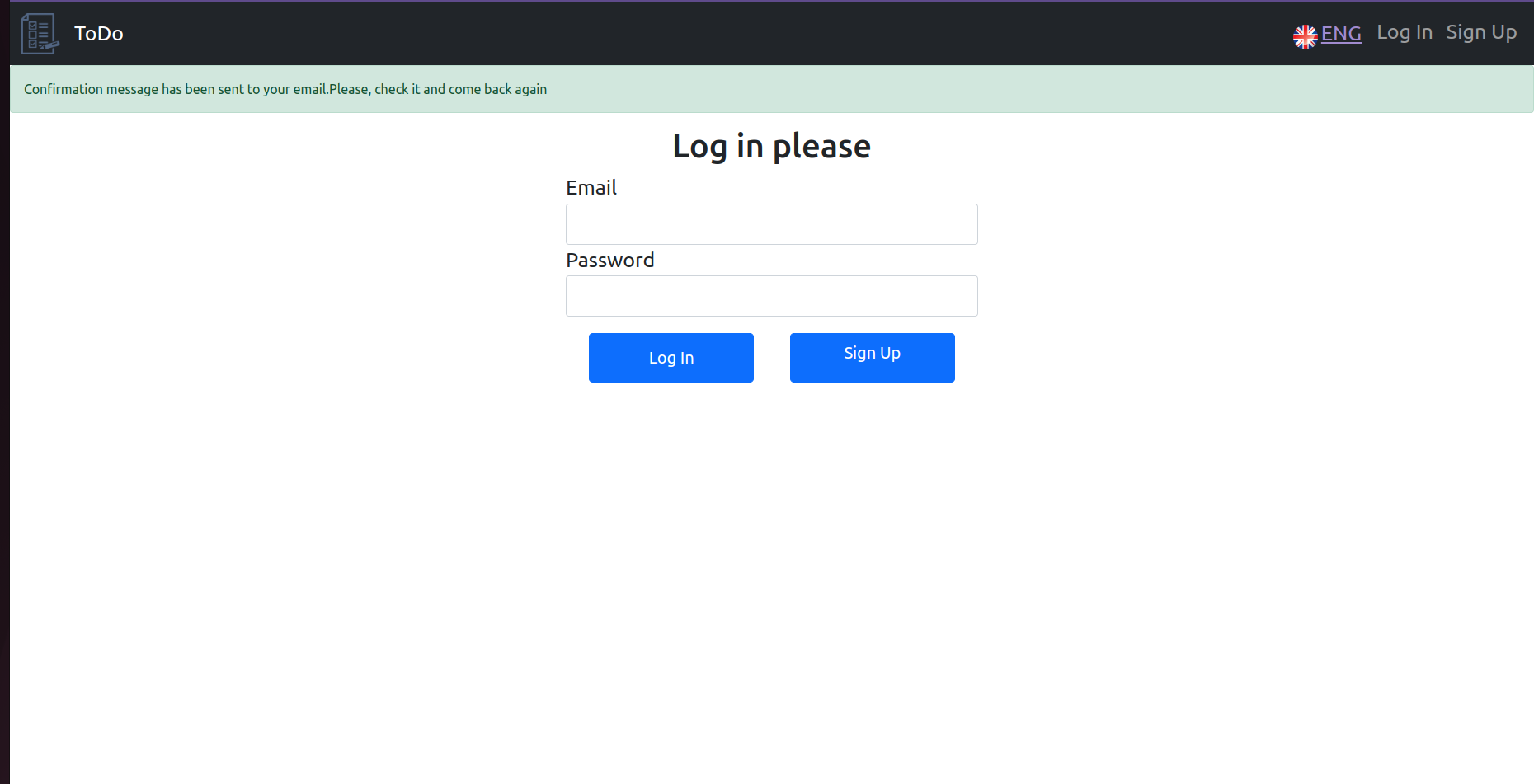
**Цель работы:**

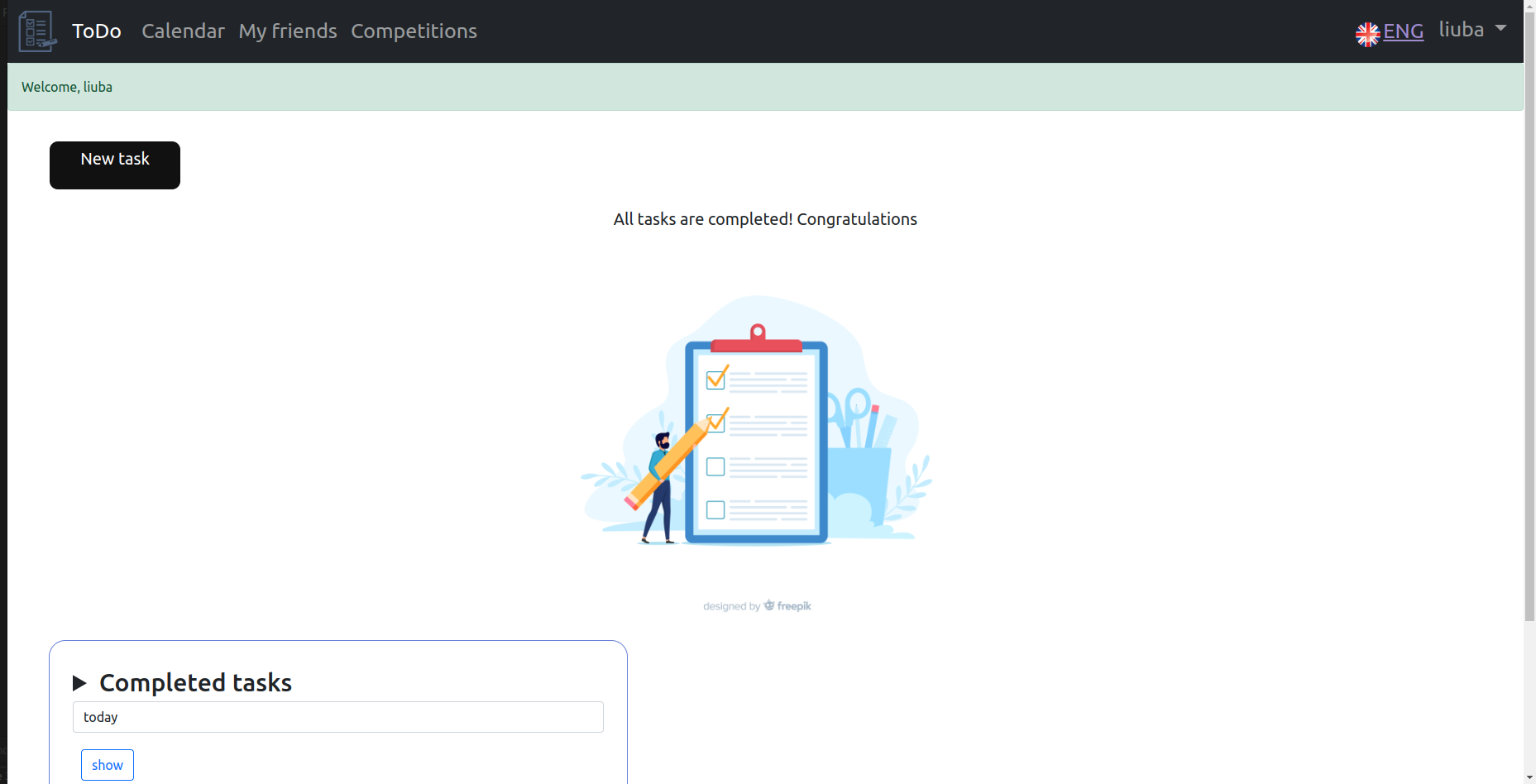
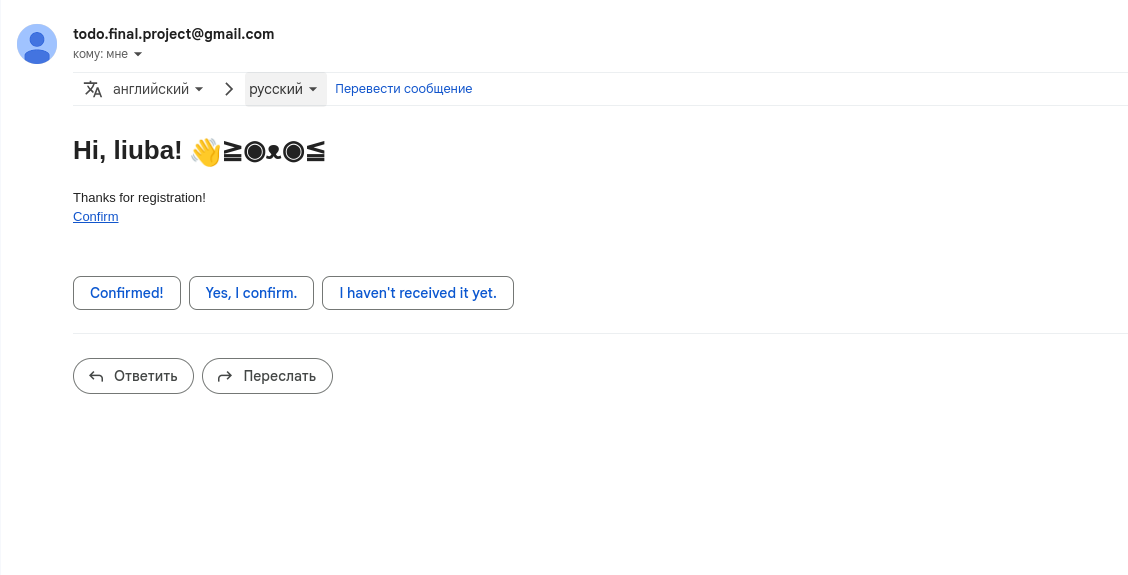
Целью зачетной работы является создание удобного, универсального планировщика с возможностью выбора карйнего срока, просмотра задач в календаре по месяцам и неделям, регулировка звуков при выполнении задач, добавление друзей и отслеживание собственного прогресса, прогресса друзей и других пользователей.

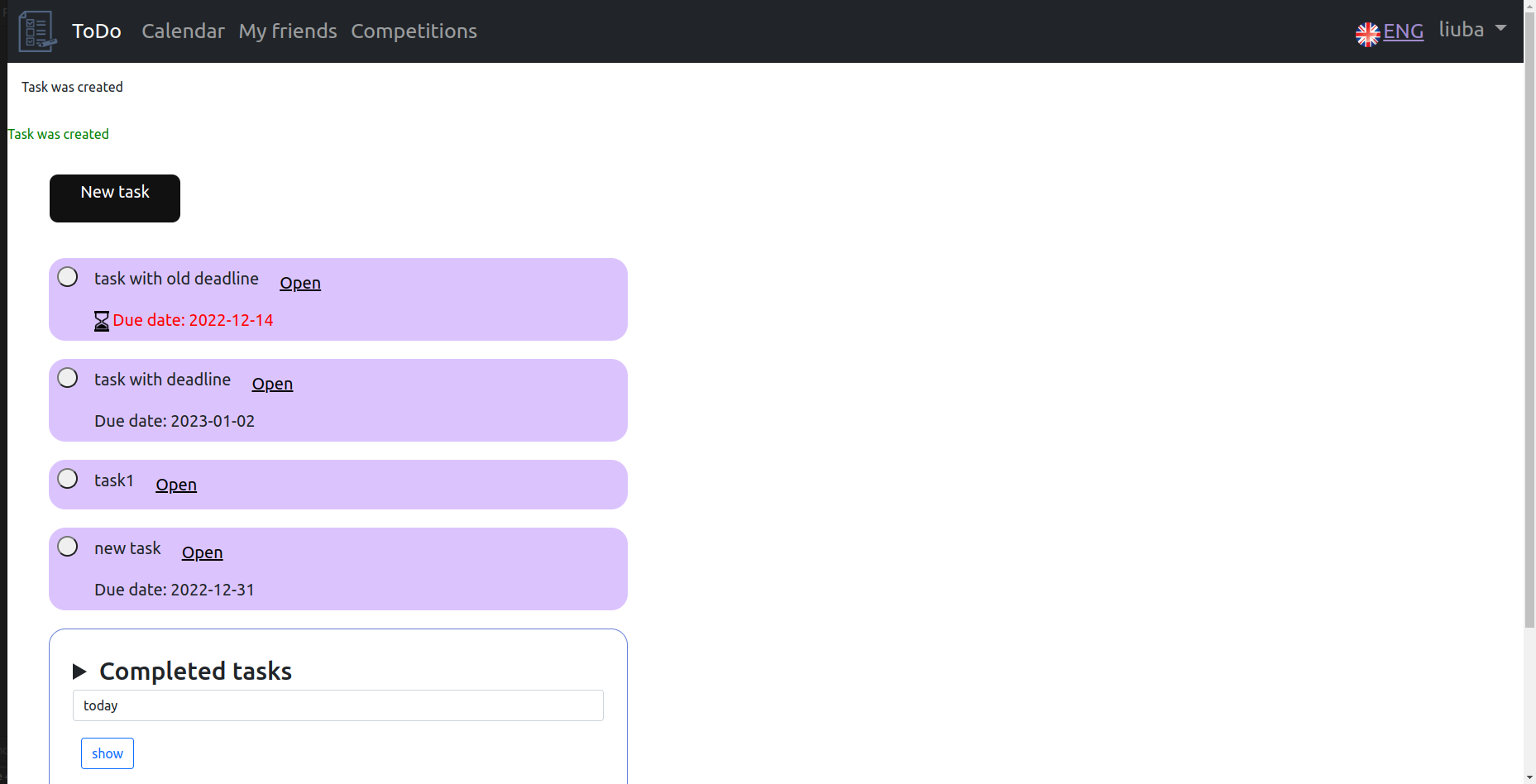
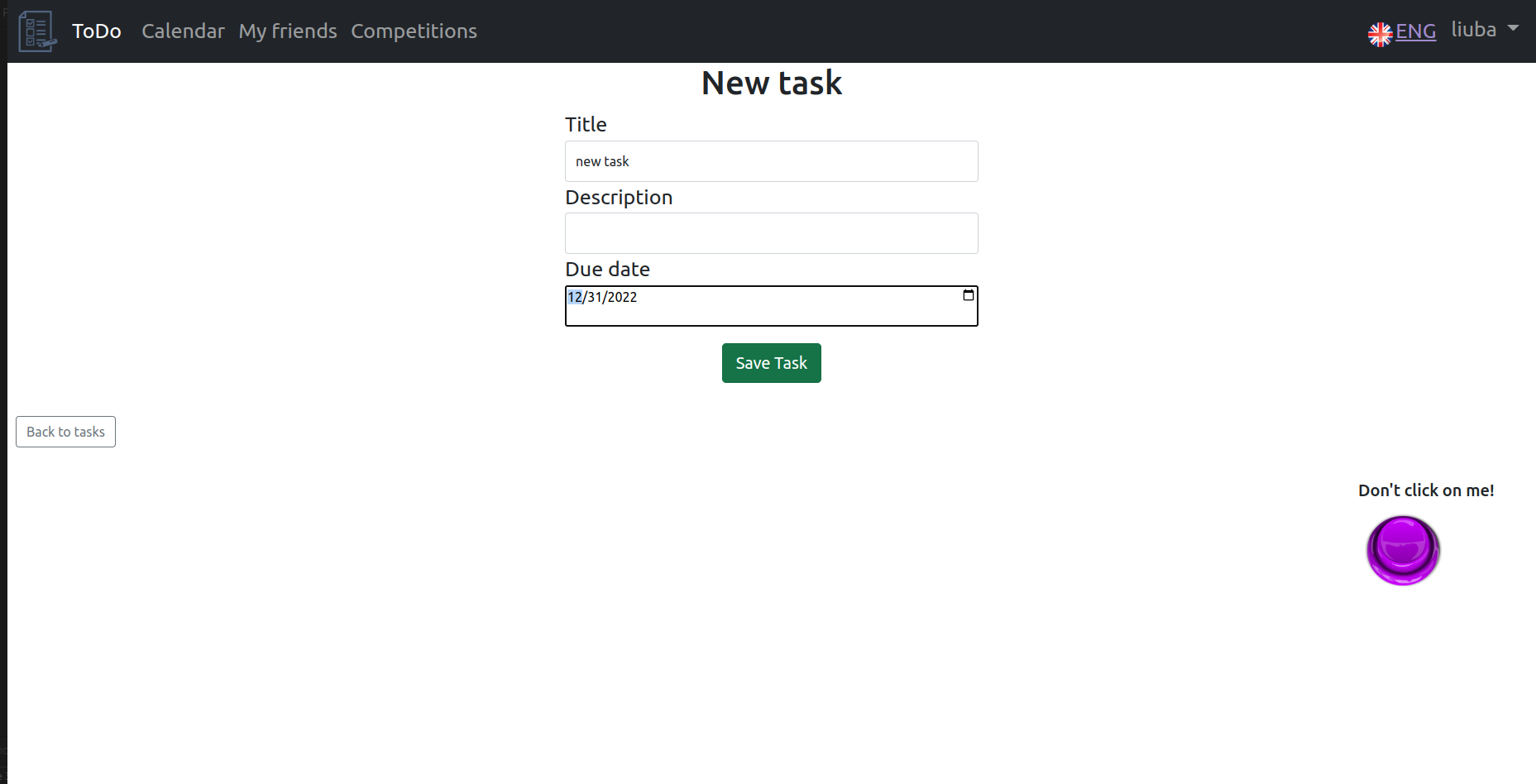
**Выполнение программы.**

**Регистрация**

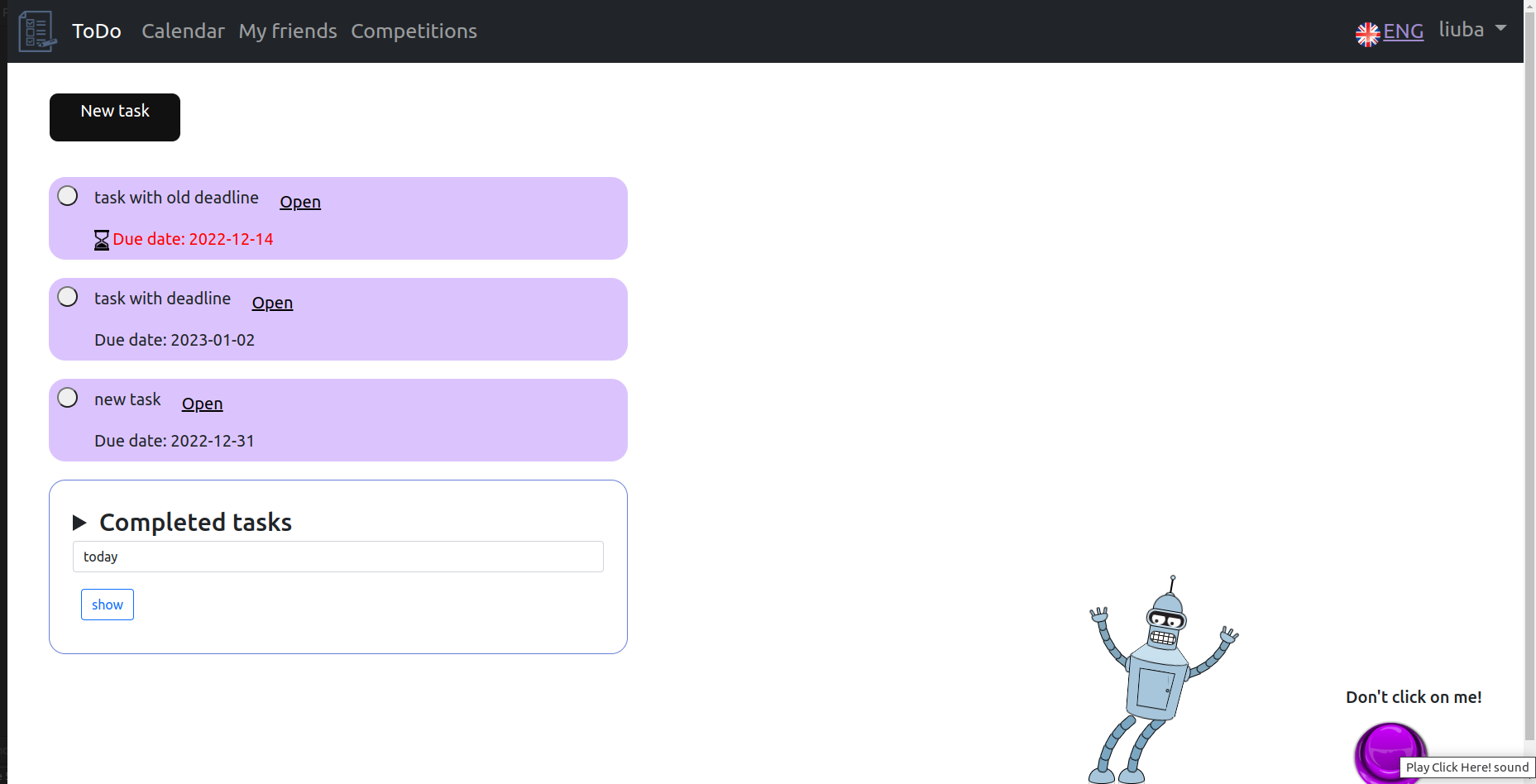
****

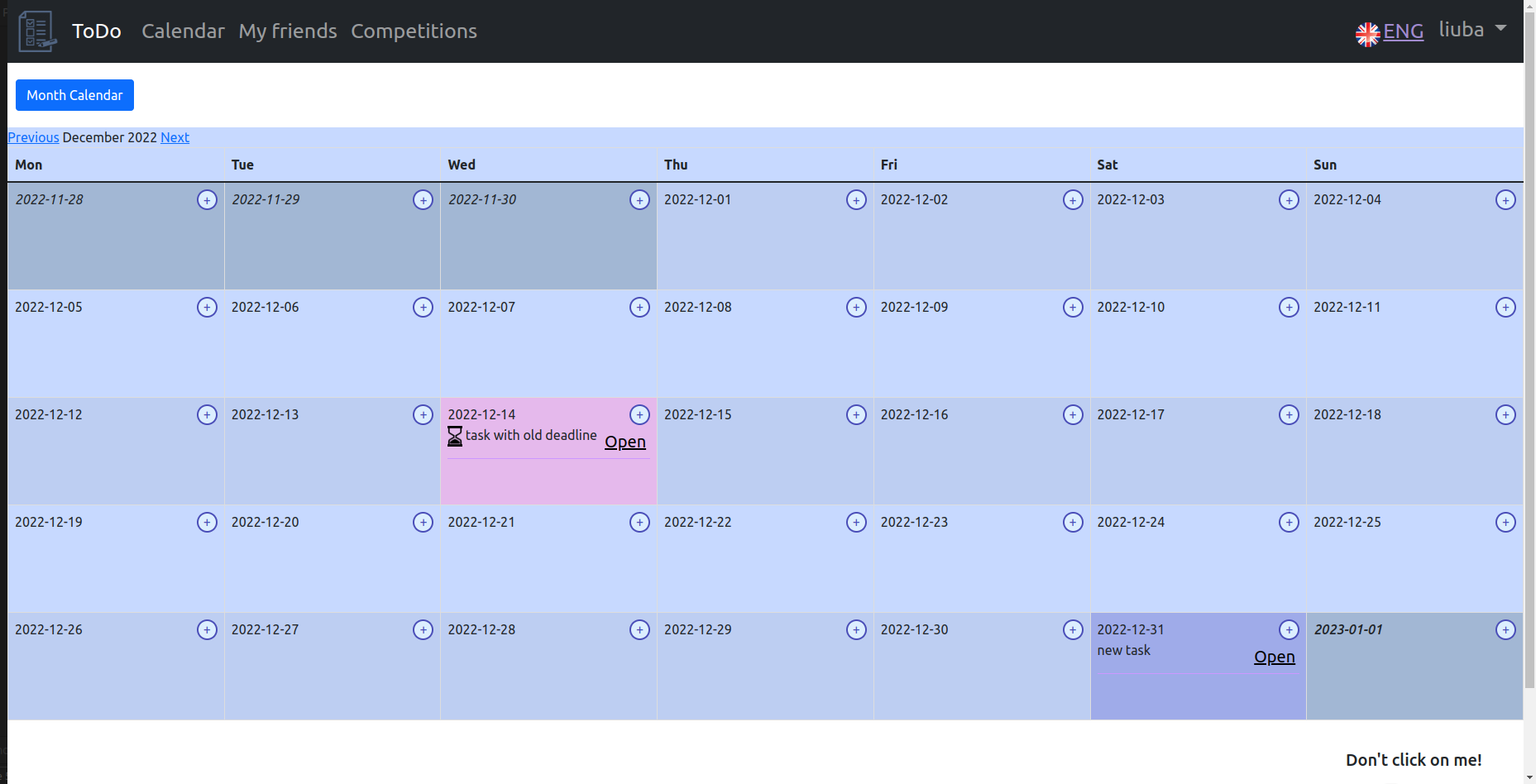
****

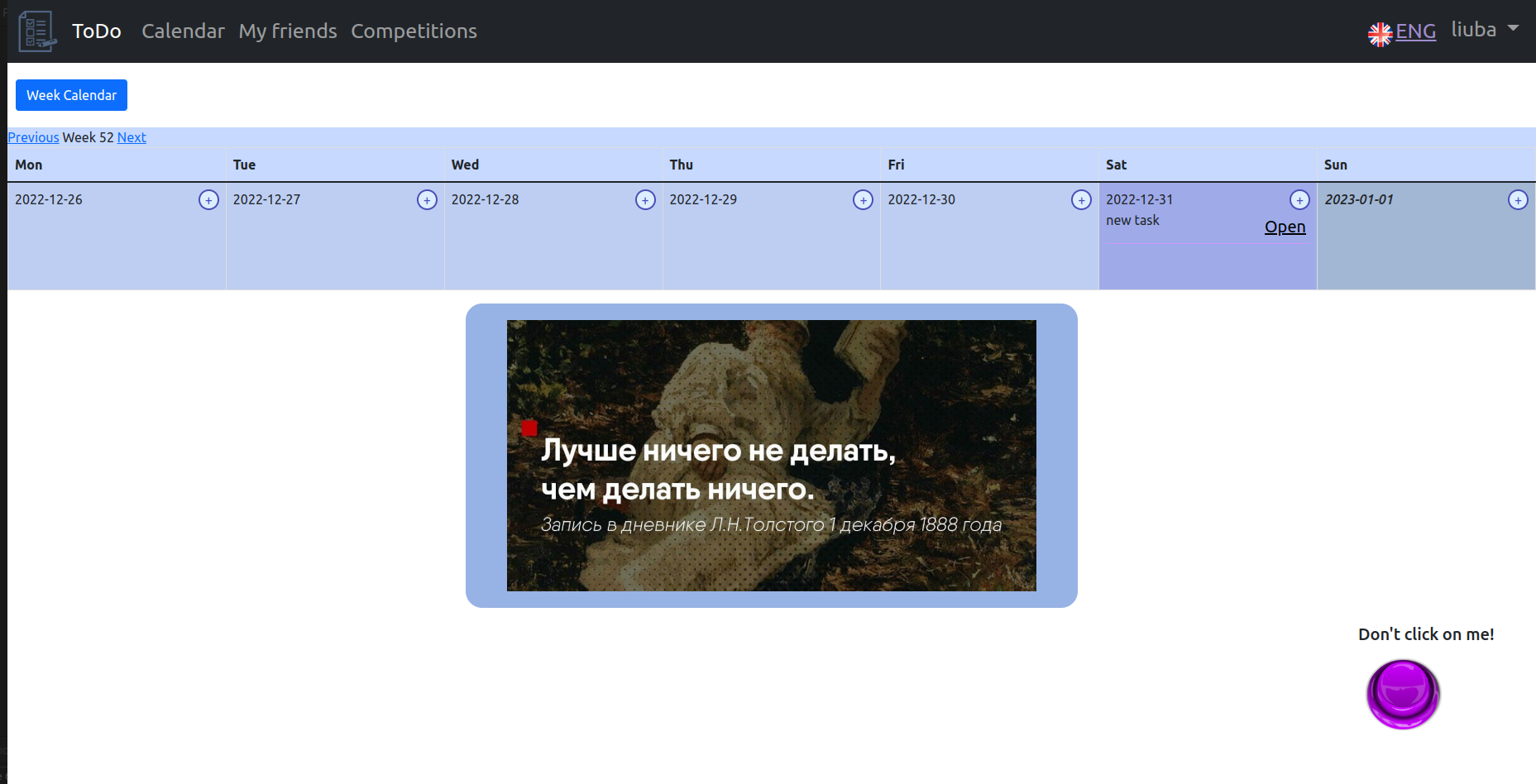
****

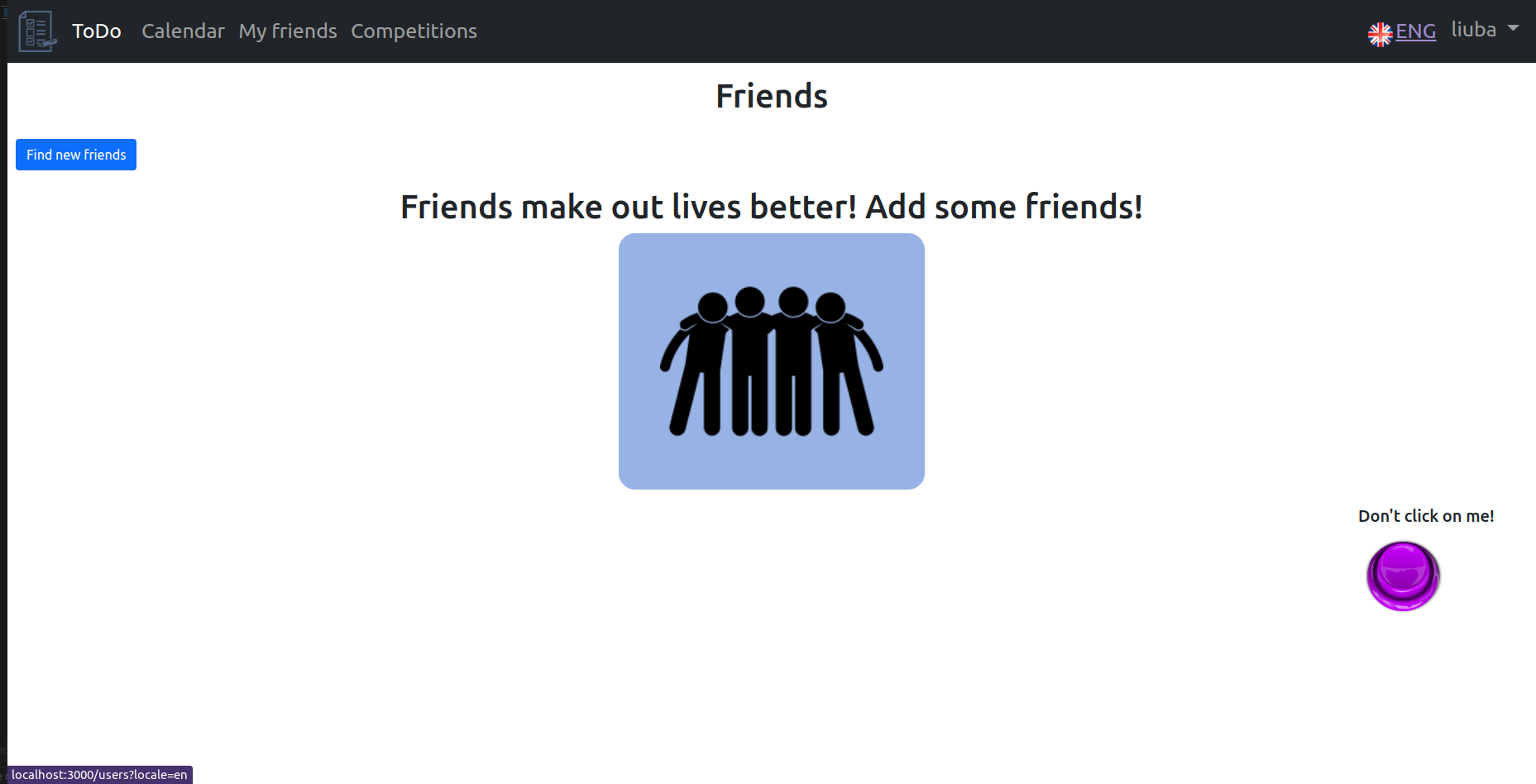
****

**При наведении появляется танцующий робот и при нажатии на кнопку играет музыка «Never gonna give you up»**

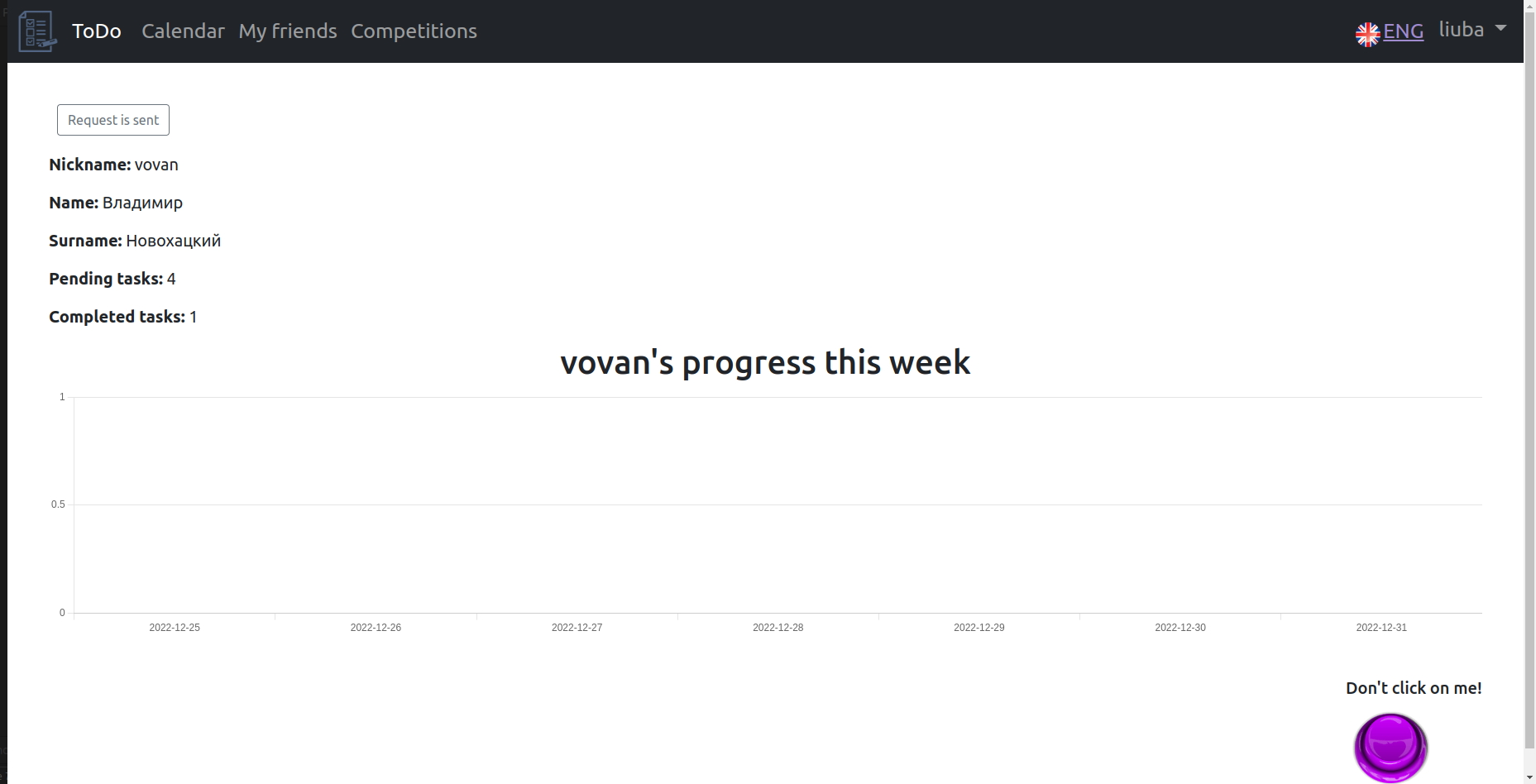
****

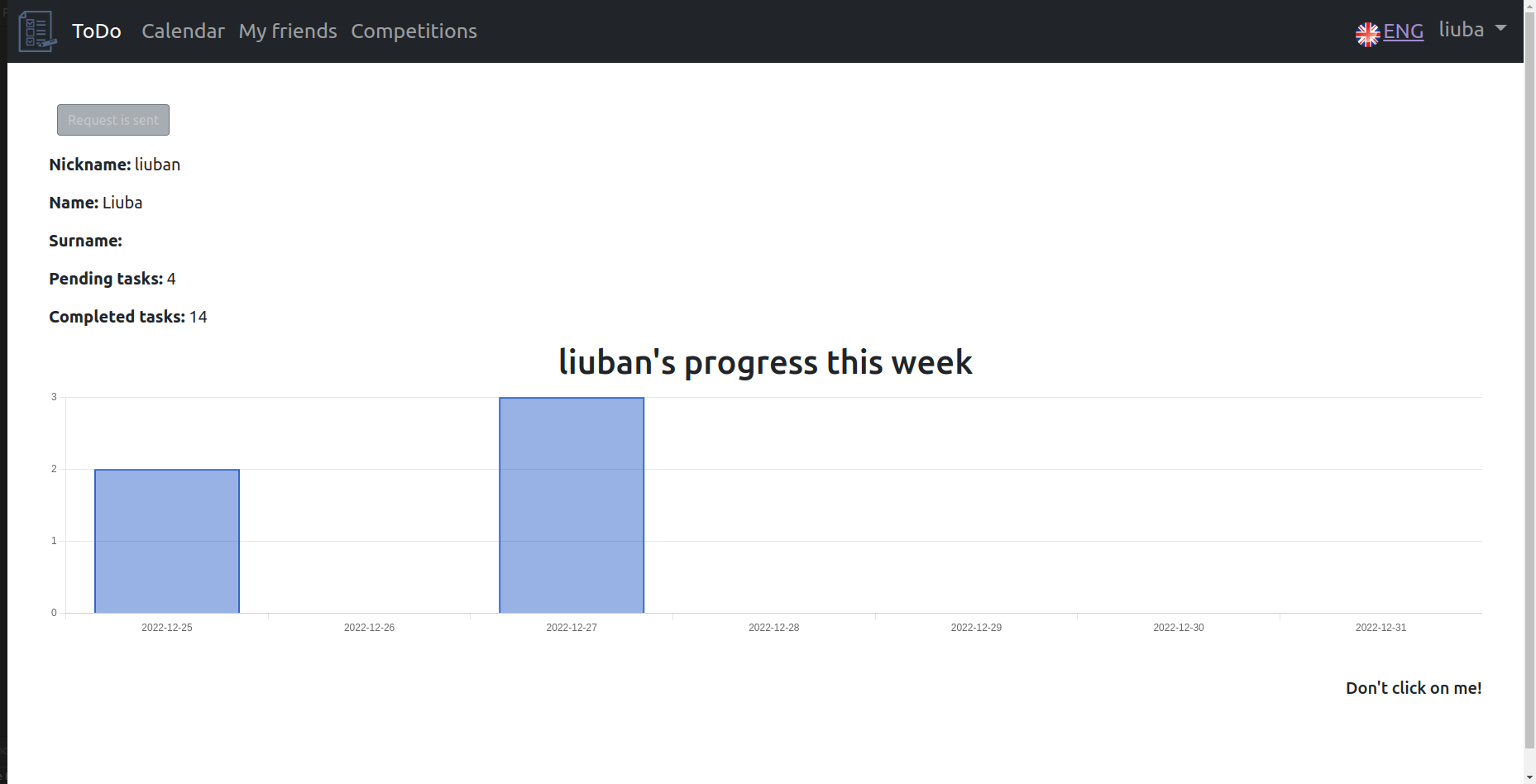
****

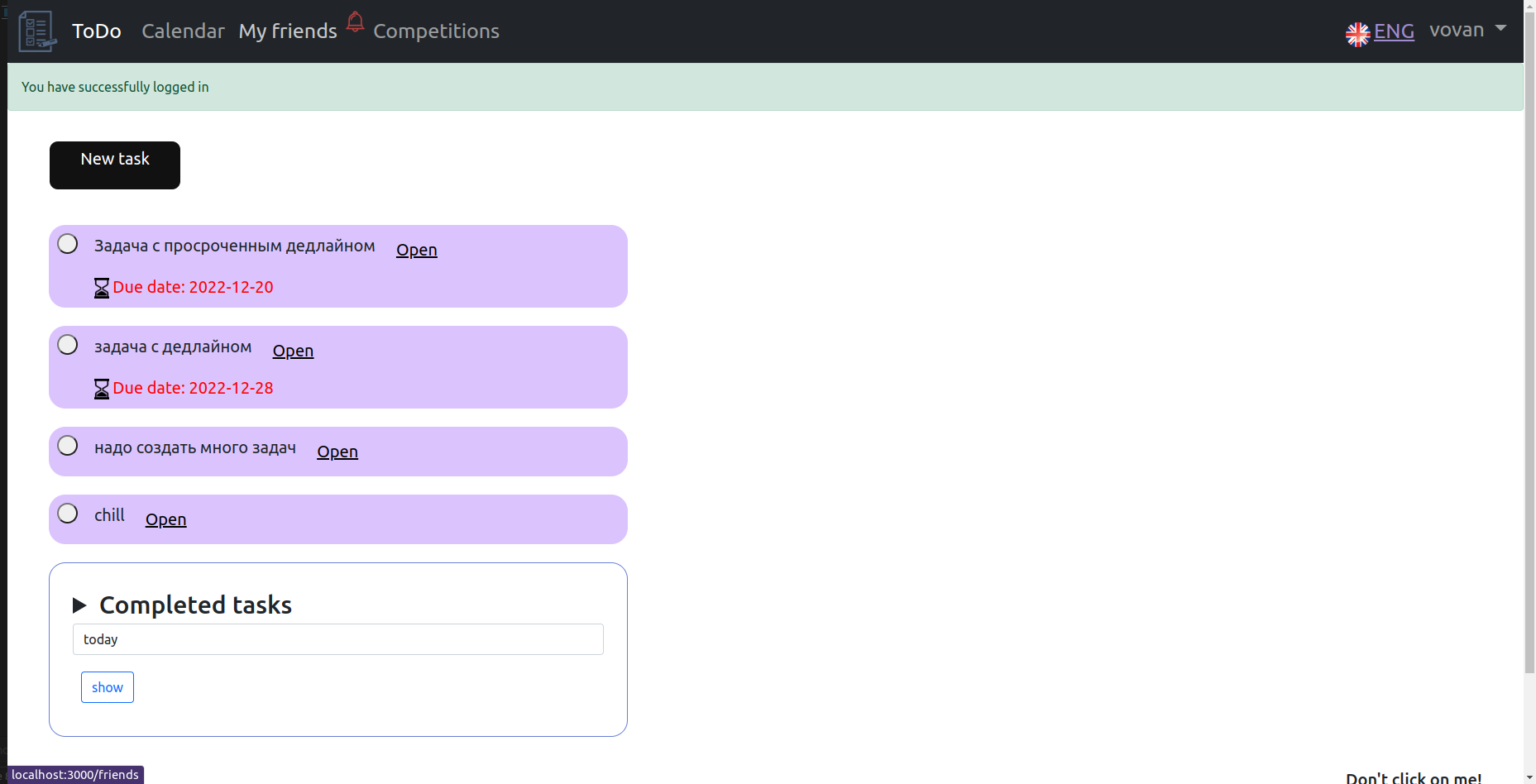
****

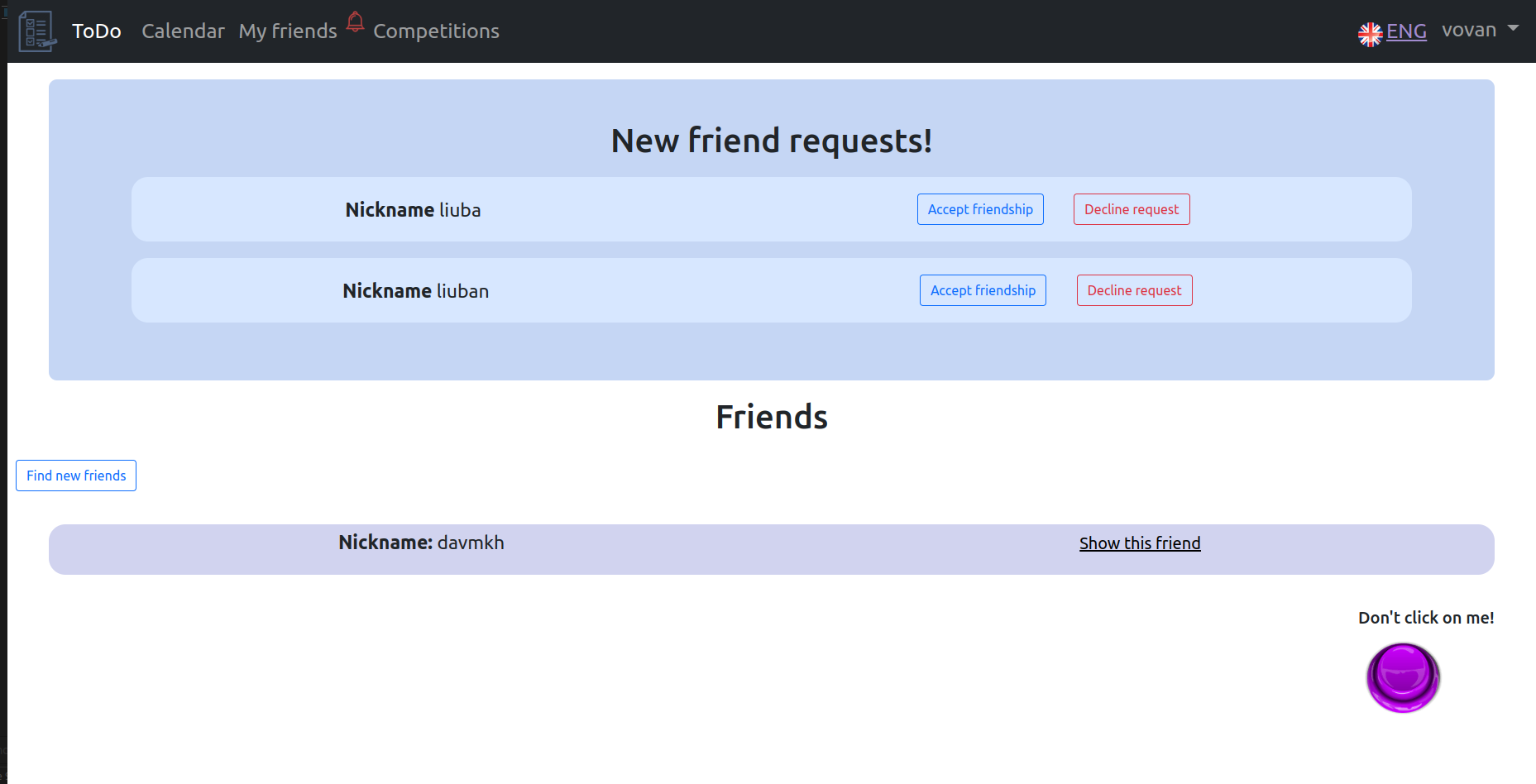
****

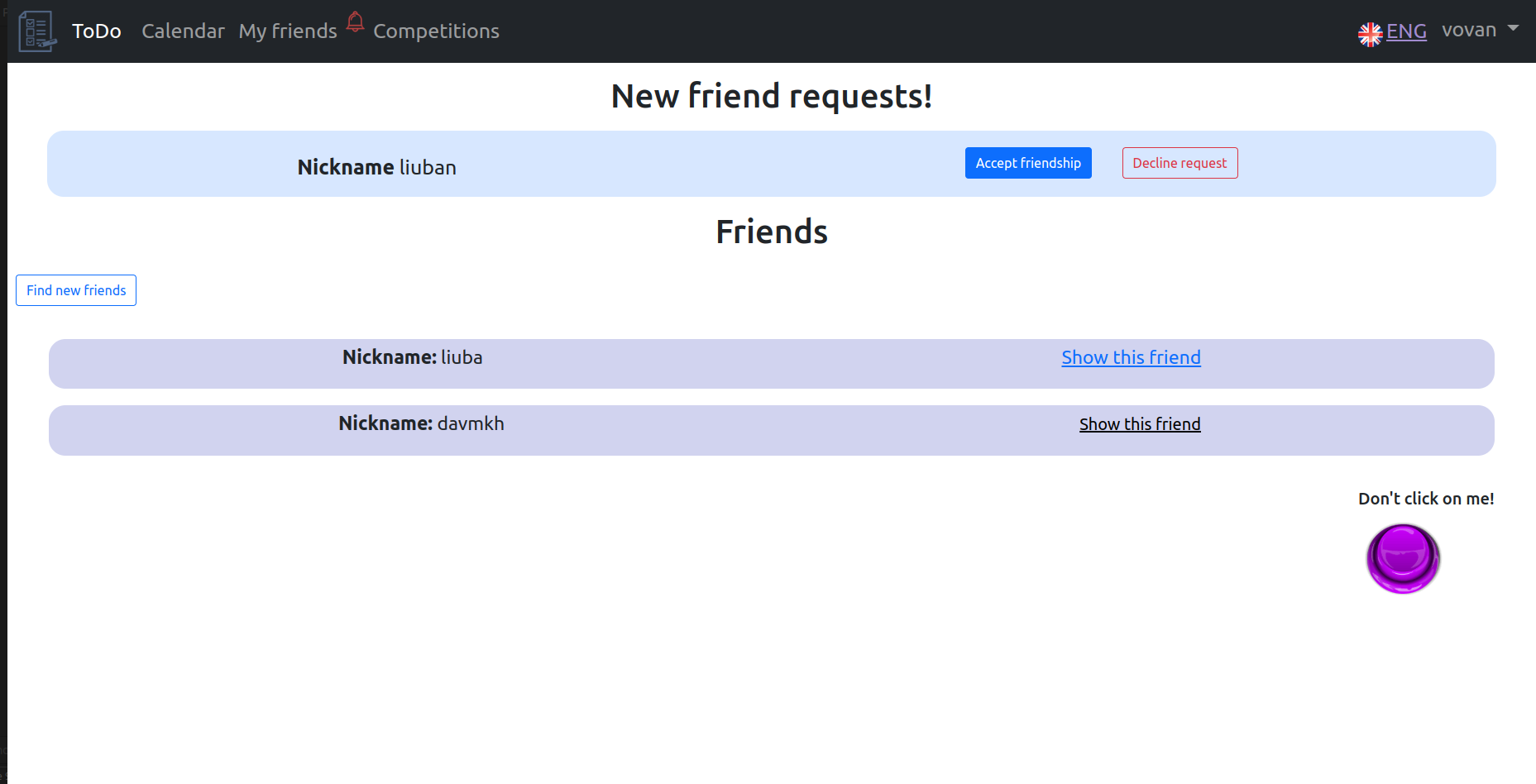
****

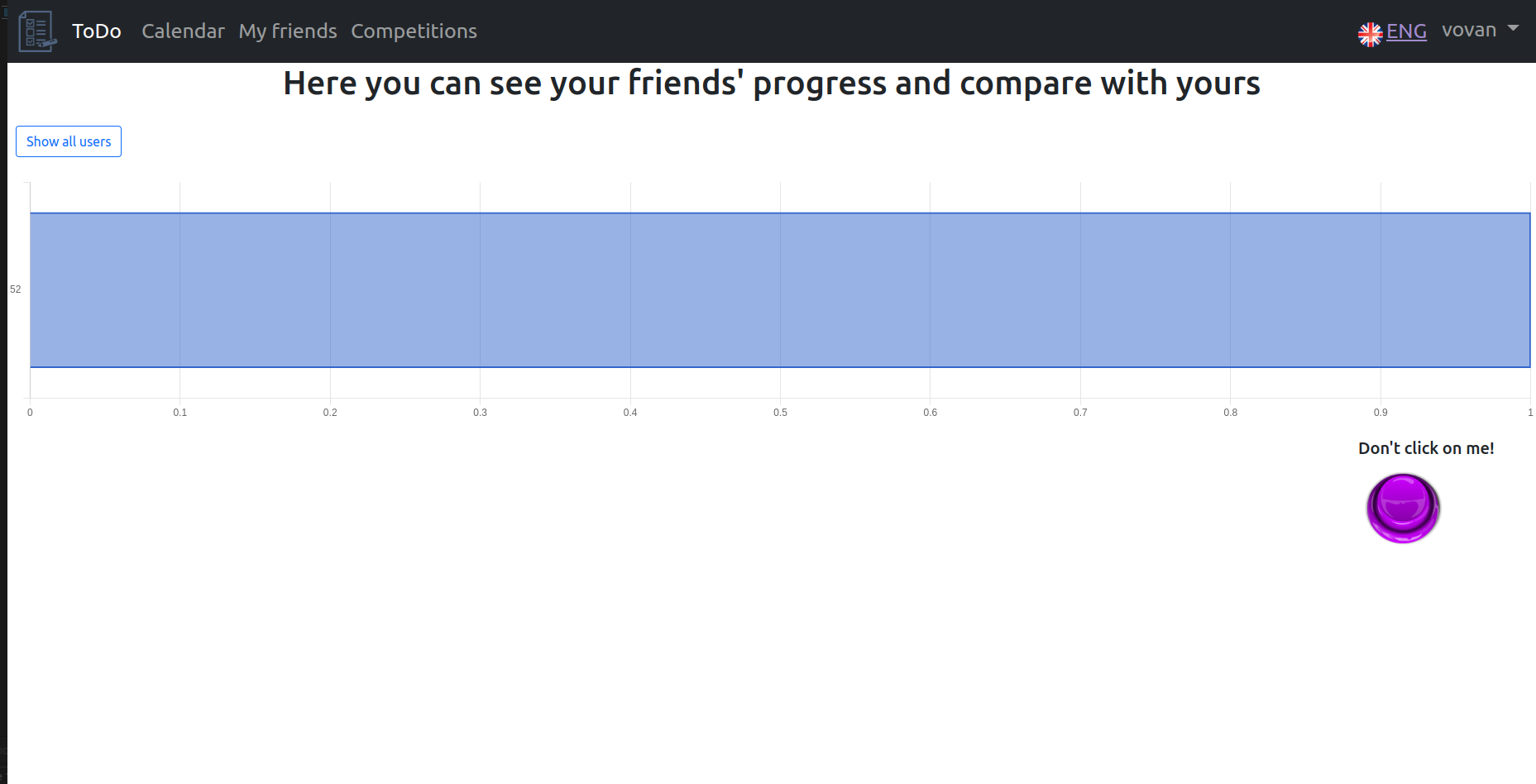
****

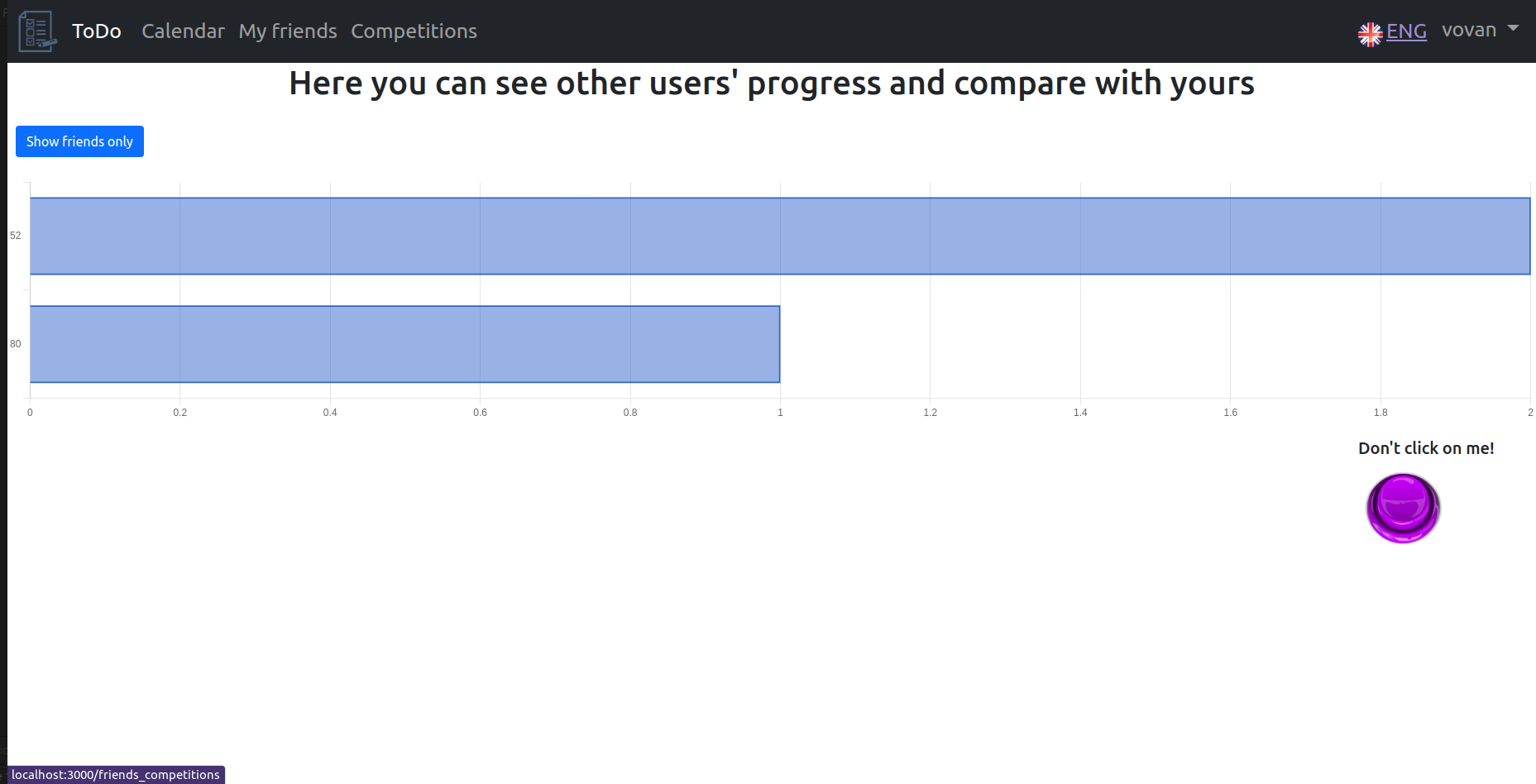
****

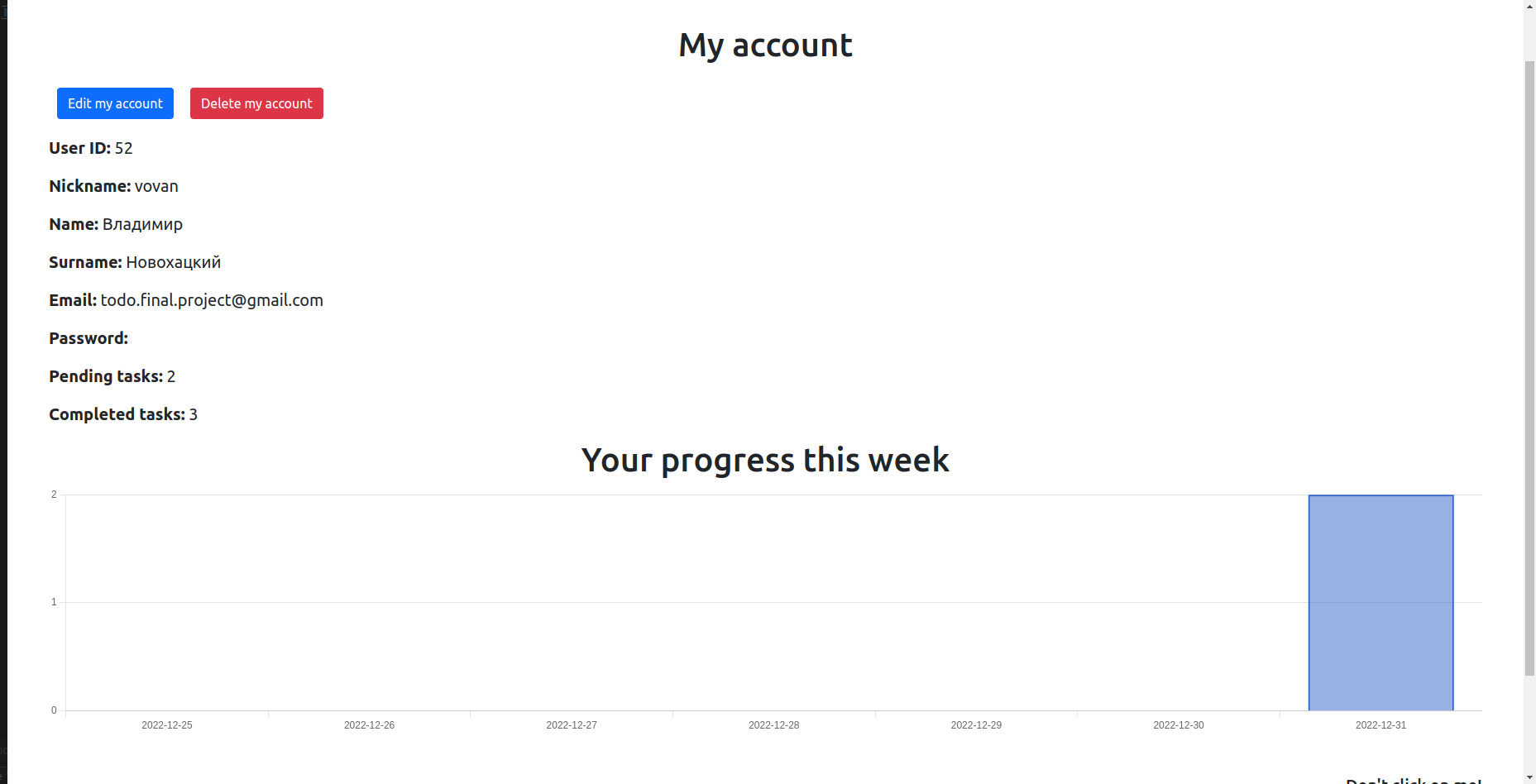
****

****

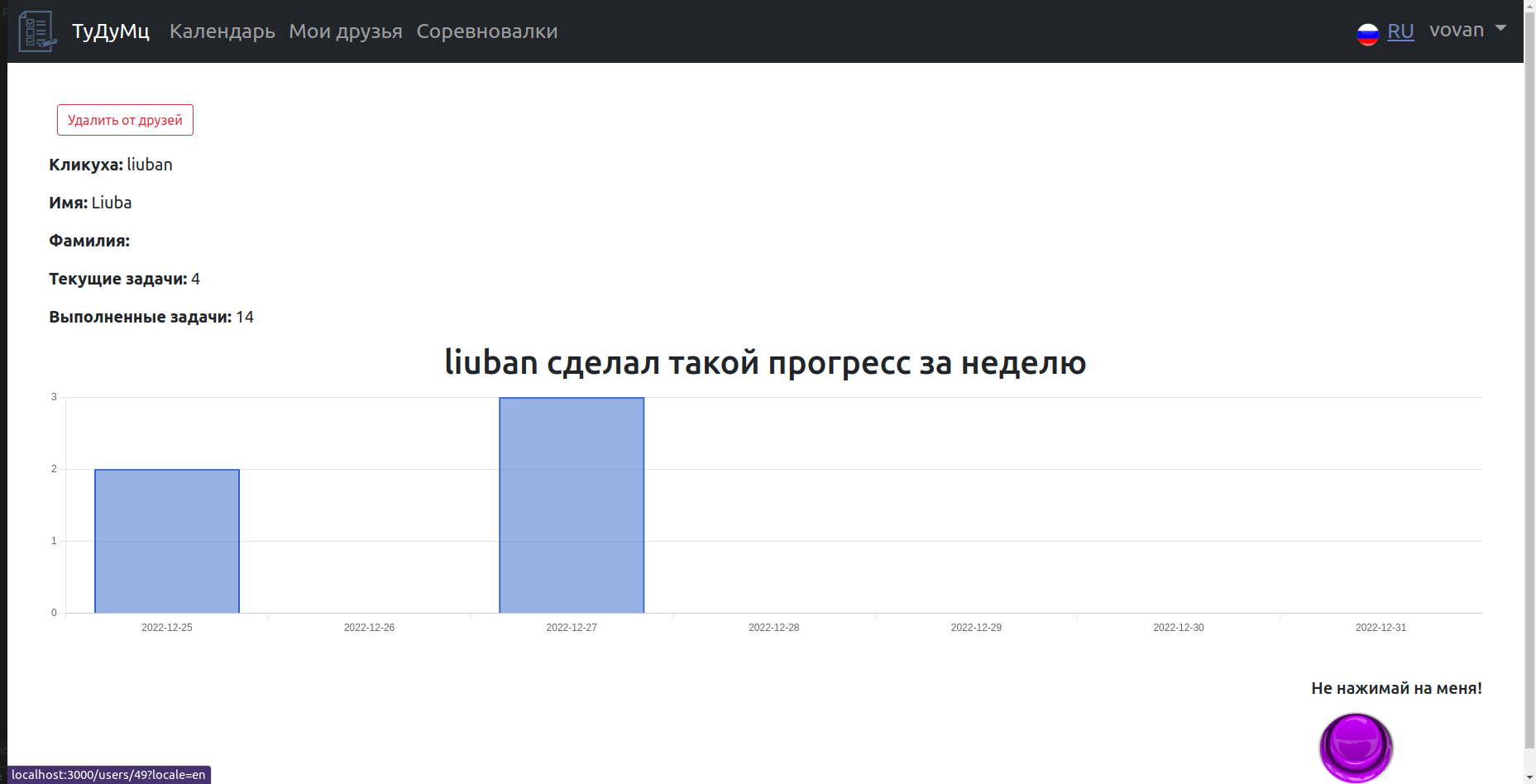
****

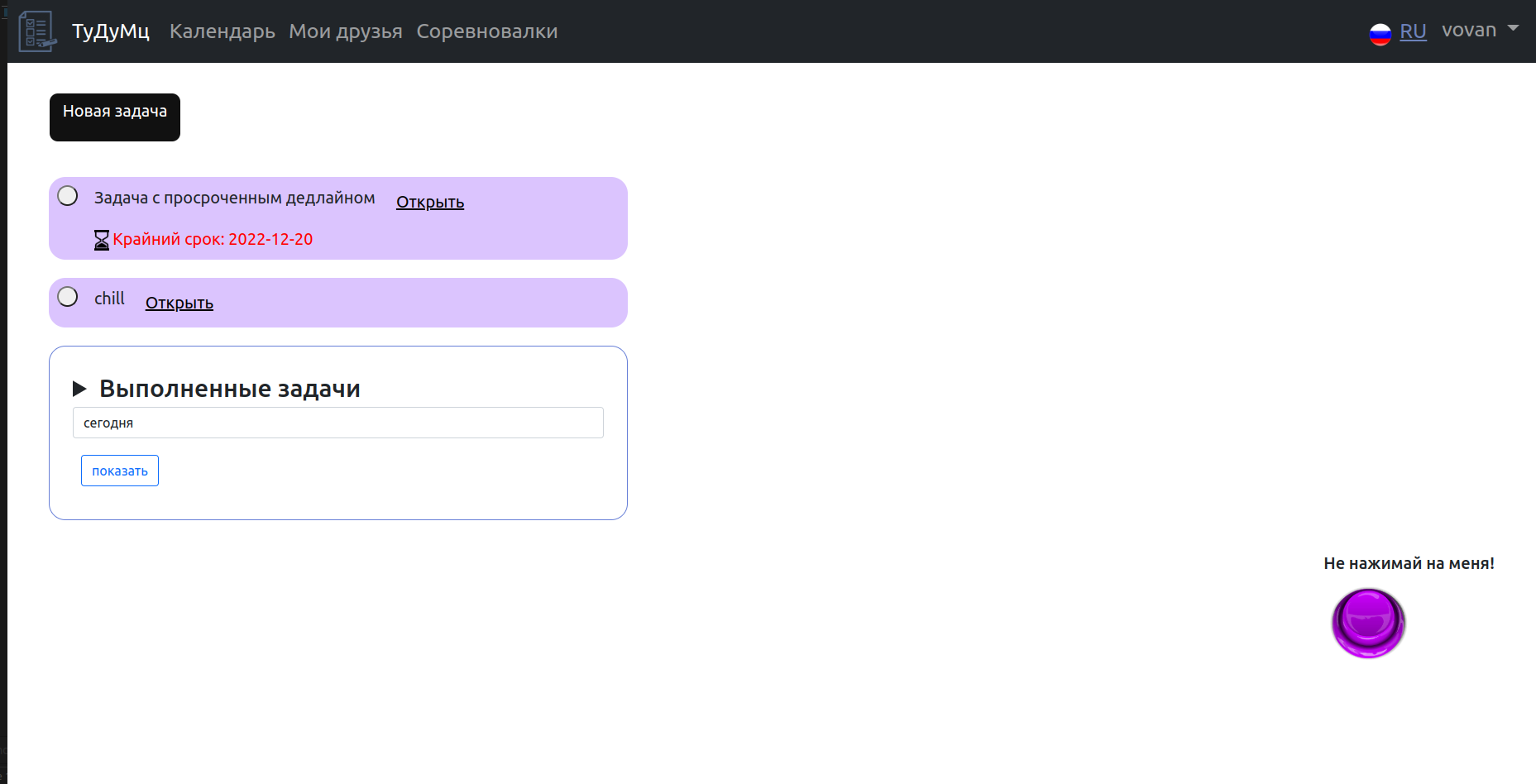
****

****

****

**Локализация:**

****

****

**Код программы.**

**app/controllers/**

**application\_controller.rb**

# frozen\_string\_literal: true

# class of controller

class ApplicationController < ActionController::Base

include ApplicationHelper

around\_action :switch\_locale

private

def switch\_locale(&action)

locale = checked\_locale || I18n.default\_locale

I18n.with\_locale locale, &action

end

def checked\_locale

locale = params[:locale]

return locale if I18n.available\_locales.map(&:to\_s).include?(locale)

end

def default\_url\_options

{ locale: I18n.locale }

end

def current\_user

@current\_user || User.find\_by(id: session[:user\_id]) if session[:user\_id].present?

end

def user\_signed\_in?

current\_user.present?

end

def autorize

return if user\_signed\_in?

flash[:warning] = t(:you\_are\_not\_logged\_in)

redirect\_to root\_path

end

def no\_autorize

return unless user\_signed\_in?

flash[:warning] = t(:you\_are\_logged\_in)

end

def app\_params

params.permit(:locale)

end

helper\_method :current\_user, :user\_signed\_in?

end

**calendars\_controller.rb**

# frozen\_string\_literal: true

# calendars

class CalendarsController < ApplicationController

# rubocop:disable Layout/LineLength

def month

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

start\_date = params.fetch(:due, Date.today).to\_date

@tasks = Task.where(due: start\_date.beginning\_of\_month.beginning\_of\_week..start\_date.end\_of\_month.end\_of\_week).where(user\_id: current\_user.id)

end

def week

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

start\_date = params.fetch(:due, Date.today).to\_date

@tasks = Task.where(due: start\_date.beginning\_of\_month.beginning\_of\_week..start\_date.end\_of\_month.end\_of\_week).where(user\_id: current\_user.id)

end

# rubocop:enable Layout/LineLength

end

**competitions\_controller.rb**

# frozen\_string\_literal: true

# competitions

class CompetitionsController < ApplicationController

include FriendsHelper

def friends

redirect\_to login\_path unless user\_signed\_in?

arr = Task.where(completed: true).select { |task| User.find(task.user\_id).is\_friend?(current\_user) || task.user\_id == current\_user.id}.select{|task| task.completed\_time.to\_date.day == Date.today.day }

@completed\_tasks = Task.where(id: arr.map(&:id)).group(:user\_id)

end

def all\_users

redirect\_to login\_path unless user\_signed\_in?

arr = Task.where(completed: true).select{|task| task.completed\_time.to\_date.day == Date.today.day }

@completed\_tasks = Task.where(id: arr.map(&:id)).group(:user\_id)

end

end

**friends\_controller.rb**

# frozen\_string\_literal: true

# friends

class FriendsController < ApplicationController

include FriendsHelper

def index

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@requests = requests

@friends = friends

end

# rubocop:disable Metrics/AbcSize

def send\_request

@requests = requests

@friend1 = User.find(friend\_params[:friend1])

@friend2 = User.find(friend\_params[:friend2])

@user = User.find(second\_user(@friend1, @friend2).id)

@possible\_friend = Friend.create!(friend\_params)

retrun unless @possible\_friend.valid?

@possible\_friend.update\_attribute(:relationship, 'request')

@possible\_friend.update\_attribute(:who\_sent\_request, @friend1.id)

end

# rubocop:enable Metrics/AbcSize

def cancell\_request

@requests = requests

@friend1 = User.find(friend\_params[:friend1])

@friend2 = User.find(friend\_params[:friend2])

@user = User.find(second\_user(@friend1, @friend2).id)

@request = find\_request(@friend1, @friend2)

@request.delete

end

def accept\_request

@requests = requests

@friend1 = User.find(friend\_params[:friend1])

@friend2 = User.find(friend\_params[:friend2])

@user = User.find(second\_user(@friend1, @friend2).id)

@request = find\_request(@friend2, @friend1)

@request.update\_attribute(:relationship, 'friendship')

@friendship = @request

end

def decline\_request

@requests = requests

@friend1 = User.find(friend\_params[:friend1])

@friend2 = User.find(friend\_params[:friend2])

@user = User.find(second\_user(@friend1, @friend2).id)

@loser = find\_request(@friend2, @friend1)

@loser.delete

end

def destroy\_friendship

@requests = requests

@friend1 = User.find(friend\_params[:friend1])

@friend2 = User.find(friend\_params[:friend2])

@user = User.find(second\_user(@friend1, @friend2).id)

@loser = find\_friend(@friend2)

@loser.delete

end

private

def friend\_params

params.permit(:friend1, :friend2)

end

end

**sessions\_controller.rb**

# frozen\_string\_literal: true

# class SessionsController

class SessionsController < ApplicationController

before\_action :no\_autorize, only: %i[new create]

before\_action :autorize, only: :destroy

def new; end

# rubocop:disable Metrics/MethodLength, Metrics/AbcSize

def create

@user = User.find\_by(email: params[:email])

if !!@user && @user.authenticate(params[:password])

if @user.email\_confirmed?

session[:user\_id] = @user.id

session[:user\_nick] = @user.nick

flash[:success] = t(:you\_have\_successfully\_logged\_in)

redirect\_to root\_path

else

flash[:error] = t(:confirm\_you\_email)

redirect\_to login\_path

end

else

flash[:error] = t(:incorrect\_password\_or\_email)

redirect\_to login\_path

end

end

# rubocop:enable Metrics/MethodLength, Metrics/AbcSize

def destroy

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

session.delete :user\_id

flash[:success] = t(:successfull\_exit)

redirect\_to root\_path

end

end

**tasks\_controller.rb**

# frozen\_string\_literal: true

# tasks

class TasksController < ApplicationController

before\_action :set\_task, only: %i[show edit update destroy]

# GET /tasks or /tasks.json

def index

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@tasks = Task.all

@option = 'today'

@tasks = Task.all.where(user\_id: current\_user.id)

@pending\_tasks = Task.where(user\_id: current\_user.id).count

end

# GET /tasks/1 or /tasks/1.json

def show; end

# GET /tasks/new

def new

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@task = Task.new

end

# GET /tasks/1/edit

def edit; end

# POST /tasks or /tasks.json

# rubocop:disable Metrics/AbcSize, Metrics/MethodLength

def create

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

if user\_signed\_in?

@task = Task.new(task\_params)

@task.user\_id = current\_user.id

end

respond\_to do |format|

if @task.save

format.html { redirect\_to root\_path, notice: t(:task\_was\_created) }

format.json { render :show, status: :created, location: @task }

else

format.html { render :new, status: :unprocessable\_entity }

# format.json { render json: @task.errors, status: :unprocessable\_entity }

end

end

end

# rubocop:enable Metrics/AbcSize, Metrics/MethodLength

# PATCH/PUT /tasks/1 or /tasks/1.json

def update

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

respond\_to do |format|

if @task.update(task\_params)

format.html { redirect\_to task\_url(@task), notice: t(:task\_was\_updated) }

# format.json { render :show, status: :ok, location: @task }

else

format.html { render :edit, status: :unprocessable\_entity }

# format.json { render json: @task.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /tasks/1 or /tasks/1.json

def destroy

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@task.destroy

respond\_to do |format|

format.html { redirect\_to tasks\_url, notice: t(:task\_was\_destroyed) }

end

end

# rubocop:disable Metrics/MethodLength, Metrics/AbcSize

def complete

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@tasks = Task.all.where(user\_id: current\_user.id)

@task = Task.find(params[:task])

@task.update\_attribute(:completed, true)

@task.update\_attribute(:completed\_time, DateTime.now)

@option = 'today'

respond\_to do |format|

format.html { render :complete } # renders `page.html.erb`

format.turbo\_stream { render :complete } # renders `page.turbo\_stream.erb`

end

end

# rubocop:enable Metrics/MethodLength, Metrics/AbcSize

# rubocop:disable Metrics/MethodLength, Metrics/AbcSize

def uncomplete

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@tasks = Task.all.where(user\_id: current\_user.id)

@task = Task.find(params[:task])

@task.update\_attribute(:completed, false)

@task.update\_attribute(:completed\_time, nil)

@option = 'today'

respond\_to do |format|

format.html { render :uncomplete } # renders `page.html.erb`

format.turbo\_stream { render :uncomplete } # renders `page.turbo\_stream.erb`

end

end

# rubocop:enable Metrics/MethodLength, Metrics/AbcSize

def choose

@tasks = Task.all.where(user\_id: current\_user.id)

redirect\_to login\_path unless user\_signed\_in?

return unless user\_signed\_in?

@option = page\_params[:option]

respond\_to do |format|

format.html { render :choose } # renders `page.html.erb`

format.turbo\_stream { render :choose } # renders `page.turbo\_stream.erb`

end

end

private

# Use callbacks to share common setup or constraints between actions.

def set\_task

@task = Task.find(params[:id])

end

# Only allow a list of trusted parameters through.

def task\_params

params.require(:task).permit(:title, :description, :due, :reminder, :reminder\_time, :completed)

end

def page\_params

params.permit(:option)

end

end

**users\_controller.rb**

# frozen\_string\_literal: true

# users

class UsersController < ApplicationController

include UsersHelper

include FriendsHelper

before\_action :set\_user, only: %i[show edit update destroy]

# before\_action :set\_color, only: %i[show]

# GET /users or /users.json

def index

redirect\_to login\_path unless user\_signed\_in?

@users = User.all

end

# GET /users/1 or /users/1.json

def show

redirect\_to login\_path unless user\_signed\_in?

@pending\_tasks = Task.where(user\_id: @user.id, completed: false)

@completed\_tasks = Task.where(user\_id: @user.id, completed: true)

end

# GET /users/new

def new

@user = User.new

end

# GET /users/1/edit

def edit; end

# POST /users or /users.json

# rubocop:disable Metrics/AbcSize

def create

@user = User.new(user\_params)

if !@user.save

flash[:error] = @user.errors.full\_messages.join('! ')

redirect\_to new\_path

else

UserMailer.registration\_confirmation(@user).deliver\_later

flash[:success] = t(:email\_sent)

redirect\_to root\_path

end

end

# rubocop:enable Metrics/AbcSize

# PATCH/PUT /users/1 or /users/1.json

def update

respond\_to do |format|

if @user.update(user\_params)

format.html { redirect\_to user\_url(@user), notice: t(:user\_was\_updated) }

format.json { render :show, status: :ok, location: @user }

else

format.html { render :edit, status: :unprocessable\_entity }

format.json { render json: @user.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /users/1 or /users/1.json

def destroy

@user.destroy

respond\_to do |format|

format.html { redirect\_to users\_url, notice: t(:user\_was\_destroyed)}

format.json { head :no\_content }

end

end

def sounds; end

def save\_sounds

@user = User.find(current\_user.id)

params.permit(:success\_sound, :failure\_sound)

@user.update\_attribute(:success\_sound, params[:success\_sound])

@user.update\_attribute(:failure\_sound, params[:failure\_sound])

redirect\_to root\_path

end

private

# Use callbacks to share common setup or constraints between actions.

def set\_user

@user = User.find(params[:id])

end

# Only allow a list of trusted parameters through.

def user\_params

params.require(:user).permit(:nick, :name, :surname, :email, :password, :password\_confirmation)

end

end

**app/helpers**

**application\_helper.rb**

# frozen\_string\_literal: true

# app helper

module ApplicationHelper

# rubocop:disable Layout/LineLength

def requests

Friend.all.where(friend1: current\_user.id,

relationship: 'request').or(Friend.all.where(friend2: current\_user.id,

relationship: 'request')).where.not(who\_sent\_request: current\_user.id)

end

# rubocop:enable Layout/LineLength

end

**friends\_helper.rb**

# frozen\_string\_literal: true

# friends helper

module FriendsHelper

# rubocop:disable Layout/LineLength

def requests

Friend.all.where(friend1: current\_user.id,

relationship: 'request').or(Friend.all.where(friend2: current\_user.id,

relationship: 'request')).where.not(who\_sent\_request: current\_user.id)

end

def friends

redirect\_to login\_path unless user\_signed\_in?

Friend.all.where(friend1: current\_user.id).or(Friend.where(friend2: current\_user.id)).where(relationship: 'friendship')

end

# rubocop:enable Layout/LineLength

def are\_friends?(user1, user2)

!Friend.where(friend1: user1.id,

friend2: user2.id).or(Friend.where(friend1: user2.id,

friend2: user1.id)).where(relationship: 'friendship').empty?

end

def find\_friend(user2)

Friend.where(friend1: current\_user.id,

friend2: user2.id).or(Friend.where(friend2: current\_user.id,

friend1: user2.id)).where(relationship: 'friendship').first

end

def find\_request(from, to)

Friend.where(friend1: from.id, friend2: to.id).or(Friend.where(friend1: to.id, friend2: from.id)).where(

relationship: 'request', who\_sent\_request: from.id

).first

end

def second\_user(user1, user2)

return user1 if user1 != current\_user

return user2 if user2 != current\_user

end

end

**users\_helper.rb**

# frozen\_string\_literal: true

# users helper

module UsersHelper

# rubocop:disable Metrics/AbcSize, Metrics/MethodLength

def confirm\_email

user = User.find\_by\_confirm\_token(params[:id])

if user

user.email\_activate

flash[:success] = "Welcome, #{user.nick}"

session[:user\_id] = user.id

session[:user\_nick] = user.nick

redirect\_to root\_path

else

flash[:error] = 'No user found with such email'

redirect\_to new\_path

end

nil

end

# rubocop:enable Metrics/AbcSize, Metrics/MethodLength

end

**javascript**

**application.js**

//Configure your import map in config/importmap.rb. Read more: https://github.com/rails/importmap-rails

import "@hotwired/turbo-rails"

import "controllers"

import "chartkick"

import "Chart.bundle"

var reminder = document.getElementById("toggle");

var reminder\_time = document.getElementById("reminder\_show");

reminder.addEventListener('change', function(){

if (this.checked) {

reminder\_time.style.visibility='visible'

}

else {

reminder\_time.style.visibility='hidden'

}

})

**mailers**

**application\_mailer.rb**

# frozen\_string\_literal: true

# application mailer

class ApplicationMailer < ActionMailer::Base

default from: ENV['TODO\_APP\_GMAIL']

layout 'mailer'

end

**user\_mailer.rb**

# frozen\_string\_literal: true

# user mailer

class UserMailer < ApplicationMailer

def registration\_confirmation(user)

@user = user

mail to: @user.email, subject: 'Email confirmation'

end

end

**models**

**friend.rb**

# frozen\_string\_literal: true

# friend

class Friend < ApplicationRecord

validates\_presence\_of :friend1, :friend2, message: I18n.t(:cannot\_be\_blank)

end

**task.rb**

# frozen\_string\_literal: true

# task

class Task < ApplicationRecord

belongs\_to :user

validates\_presence\_of :title, message: I18n.t(:cannot\_be\_blank)

# actually it returns due date, but i need this variable to be

# name as start\_time cuz i want to use task model as event

def start\_time

due

end

end

**user.rb**

# frozen\_string\_literal: true

# user

class User < ApplicationRecord

has\_secure\_password

validates\_uniqueness\_of :nick, message: I18n.t(:already\_exists)

validates\_uniqueness\_of :email, message: I18n.t(:already\_exists)

validates\_presence\_of :nick, message: I18n.t(:cannot\_be\_blank)

validates\_presence\_of :email, message: I18n.t(:cannot\_be\_blank), format: { with: URI::MailTo::EMAIL\_REGEXP }

validates\_presence\_of :password, message: I18n.t(:cannot\_be\_blank)

validates\_presence\_of :password\_confirmation, message: I18n.t(:cannot\_be\_blank)

before\_create :confirmation\_token

def email\_activate

self.email\_confirmed = true

self.confirm\_token = nil

save!(validate: false)

end

# rubocop:disable Naming/PredicateName

def is\_friend?(user)

!Friend.where(friend1: id,

friend2: user.id).or(Friend.where(friend1: user.id,

friend2: id)).where(relationship: 'friendship').empty?

end

# rubocop:enable Naming/PredicateName

private

def confirmation\_token

self.confirm\_token = SecureRandom.urlsafe\_base64.to\_s if confirm\_token.blank?

end

end

**views/calendars**

**month.html.erb**

<%=link\_to t(:month\_calendar), week\_path, type: "button", class: 'btn btn-primary', id: 'month-calendar'%>

<%= month\_calendar(events: @tasks) do |date, task| %>

<div style="display: flex;justify-content:space-between">

<%= date %>

<button type="button" class="calendar\_button" data-bs-toggle="modal" data-bs-target="#new\_task">+</button>

</div>

<% task.each do |task| %>

<div class = "task\_for\_calendar">

<% if task.completed %>

<div style="text-decoration: line-through">

<%= task.title %>

</div>

<%else%>

<div>

<%if date < Date.today()%>

<img src="<%=asset\_path('over.png')%>" style="height:25px">

<%end%>

<%= task.title %>

</div>

<%end%>

<a class="link" data-bs-toggle="modal" data-bs-target="#month<%= dom\_id task %>"><%=t(:open\_task)%></a>

<div class="modal fade" id="month<%= dom\_id task %>" tabindex="-1" aria-labelledby="open\_label" aria-hidden="true">

<div class="modal-dialog modal-xl">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="open\_label"><%= t(:task\_label)%></h4>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

</div>

<div class="modal-body">

<%= render "tasks/task\_show", task: task %>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal"><%=t(:close\_modal)%></button>

</div>

</div>

</div>

</div>

</div>

<% end %>

<% end %>

**week.html.erb**

<%=link\_to t(:week\_calendar), month\_path, type: "button", class: 'btn btn-primary'%>

<%= week\_calendar(events: @tasks) do |date, task| %>

<div style="display: flex;justify-content:space-between">

<%= date %>

<button type="button" class="calendar\_button" data-bs-toggle="modal" data-bs-target="#new\_task">+</button>

</div>

<% task.each do |task| %>

<div class = "task\_for\_calendar">

<% if task.completed %>

<div style="text-decoration: line-through">

<%= task.title %>

</div>

<%else%>

<div>

<%if date < Date.today()%>

<img src="<%=asset\_path('over.png')%>" style="height:25px">

<%end%>

<%= task.title %>

</div>

<%end%>

<a class="link" data-bs-toggle="modal" data-bs-target="#month<%= dom\_id task %>"><%=t(:open\_task)%></a>

<div class="modal fade" id="month<%= dom\_id task %>" tabindex="-1" aria-labelledby="open\_label" aria-hidden="true">

<div class="modal-dialog modal-xl">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="open\_label"><%= t(:task\_label)%></h4>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

</div>

<div class="modal-body">

<%= render "tasks/task\_show", task: task %>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal"><%=t(:close\_modal)%></button>

</div>

</div>

</div>

</div>

</div>

<% end %>

<% end %>

<img src = "<%= image\_path('todonothing.png') %>" alt= "Tolstoy" id= "Tolstoy">

**views/competitions**

**all\_users.html.erb**

<h1><%=t(:all\_users\_competitions)%></h1>

<%=link\_to t(:compete\_with\_friends\_button), 'friends\_competitions', class: 'btn btn-outline-primary', id: "compete\_with\_friends"%>

<%= bar\_chart @completed\_tasks.count%>

**friends.html.erb**

<h1><%=t(:all\_friends\_competitions)%></h1>

<%=link\_to t(:compete\_with\_all\_users\_button), 'all\_competitions', class: 'btn btn-outline-primary', id: "compete\_with\_all"%>

<%= bar\_chart @completed\_tasks.count%>

**views/friends**

**\_accept\_request.html.erb**

<%= button\_to t(:accept\_friendship\_button), accept\_request\_path(friend1: current\_user.id, friend2: @user.id), class: 'btn btn-outline-primary' %>

**\_add\_as\_friends.html.erb**

<%= button\_to t(:add\_as\_friend\_button), send\_request\_path(friend1: current\_user.id, friend2: @user.id), class: 'btn btn-outline-primary' %>

**\_decline\_request.html.erb**

<%= button\_to t(:decline\_request\_button), decline\_request\_path(friend1: current\_user.id, friend2: @user.id), class: 'btn btn-outline-danger' %>

**\_destroy\_friendship.html.erb**

<%= button\_to t(:destroy\_friendship), destroy\_friendship\_path(friend2: @user.id, friend1: current\_user.id), class: 'btn btn-outline-danger' %>

**\_friend.html.erb**

<div id="<%= dom\_id @friendship %>" class = "friend">

<% @user = User.find(second\_user(@friendship.friend1, @friendship.friend2))%>

<p>

<strong><%=t(:nick)%>:</strong>

<%= @user.nick %>

</p>

<p>

<%= link\_to t(:show\_this\_friend), @user%>

</p>

</div>

**\_request\_is\_sent.html.erb**

<%= button\_to t(:request\_is\_sent\_button), cancell\_request\_path(friend2: @user.id, friend1: current\_user.id), class: 'btn btn-outline-secondary'%>

**\_requests.html.erb**

<%if !@requests.empty?%>

<h1><%= t(:new\_friend\_requests)%></h1>

<% @requests.each do |request| %>

<div id="<%= dom\_id request %>" class = "request">

<% @user = User.find(request.who\_sent\_request)%>

<p>

<strong><%=t(:nick)%></strong>

<%= @user.nick %>

</p>

<div class = "buttons\_near">

<%= render "friends/accept\_request", locals: {friend2: @user.id, friend1: current\_user.id} %>

<%= render "friends/decline\_request", locals: {friend2: @user.id, friend1: current\_user.id} %>

</div>

</div>

<% end %>

<%end%>

**accept\_request.turbo\_stream.erb**

<%= turbo\_stream.prepend 'friends', partial: 'friend', locals: {friend2: @user.id, friend1: current\_user.id} %>

<%= turbo\_stream.remove "make\_friends" %>

<%= turbo\_stream.update "requests", partial: "requests", locals: {friend2: @user.id, friend1: current\_user.id}%>

**cancell\_request.turbo\_stream.erb**

<%= turbo\_stream.update 'relationship\_button', partial: 'add\_as\_friend'%>

**decline\_request.turbo\_stream.erb**

<%= turbo\_stream.update "requests", partial: "requests"%>

**destroy\_friendship.turbo\_stream.erb**

<%= turbo\_stream.update 'relationship\_button', partial: 'add\_as\_friend'%>

**index.html.erb**

<p style="color: green"><%= notice %></p>

<%if !@requests.empty?%>

<div id="requests">

<div id = "requests\_for\_css">

<h1><%=t(:new\_friend\_requests)%></h1>

<% @requests.each do |request| %>

<div id="<%= dom\_id request %>" class = "request">

<% @user = User.find(request.who\_sent\_request)%>

<p>

<strong><%=t(:nick)%></strong>

<%= @user.nick %>

</p>

<div class = "buttons\_near">

<%= render "friends/accept\_request", locals: {friend2: @user.id, friend1: current\_user.id} %>

<%= render "friends/decline\_request", locals: {friend2: @user.id, friend1: current\_user.id} %>

</div>

</div>

<% end %>

</div>

</div>

<%end%>

<h1><%=t(:friends)%></h1>

<%=link\_to t(:find\_new\_friends), users\_path, class: 'btn btn-outline-primary'%>

<%if @friends.empty?%>

<div id = "make\_friends">

<h1><%=t(:make\_friends)%></h1>

<a href="<%=users\_path%>">

<img src = "<%= image\_path('4friends.png') %>" alt= "add friends" id= "add\_friends" >

</a>

</div>

<%end%>

<div id="friends">

<% @friends.each do |friend| %>

<div id="<%= dom\_id friend %>" class = "friend">

<% @user = User.find(second\_user(User.find(friend.friend1), User.find(friend.friend2)).id)%>

<%=%>

<p>

<strong><%=t(:nick)%>: </strong>

<%= @user.nick %>

</p>

<p>

<%= link\_to t(:show\_this\_friend), @user, class: 'link'%>

</p>

</div>

<% end %>

</div>

**send\_request.turbo\_stream.erb**

<%= turbo\_stream.update 'relationship\_button', partial: 'request\_is\_sent'%>

**views/layouts**

**application.html.erb**

<!DOCTYPE html>

<html>

<head>

<title>ToDoList</title>

<meta name="viewport" content="width=device-width,initial-scale=1">

<%= csrf\_meta\_tags %>

<%= csp\_meta\_tag %>

<%= stylesheet\_link\_tag "application", "data-turbo-track": "reload" %>

<%= javascript\_importmap\_tags %>

<%= javascript\_include\_tag "//www.google.com/jsapi"%>

<%= javascript\_include\_tag "application", "chartkick" %>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-uWxY/CJNBR+1zjPWmfnSnVxwRheevXITnMqoEIeG1LJrdI0GlVs/9cVSyPYXdcSF" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-kQtW33rZJAHjgefvhyyzcGF3C5TFyBQBA13V1RKPf4uH+bwyzQxZ6CmMZHmNBEfJ" crossorigin="anonymous"></script>

</head>

<body>

<header>

<nav class="navbar navbar-expand-xl navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="/"><img src="<%=asset\_path('icon.jpg')%>" style="height:50px"></a>

<a class="navbar-brand" href="/"><%=t(:todo)%></a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavDropdown">

<%if user\_signed\_in?%>

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="/month"><%=t(:calendar)%></a>

</li>

<li class="nav-item" style="display:flex;">

<a class="nav-link" href="/friends"><%=t(:my\_friends)%></a>

<%@requests = requests%>

<%if !@requests.empty?%>

<img src="<%=asset\_path('bell.png')%>" id = "red\_bell">

<%end%>

</li>

<li class="nav-item">

<a class="nav-link" href="/friends\_competitions"><%=t(:competitions)%></a>

</li>

</ul>

<%end%>

<div class="collapse navbar-collapse justify-content-end" id="navbarNav">

<ul class="navbar-nav mr-auto">

<li id = "locals">

<% if I18n.locale == :ru %>

<img src="<%=asset\_path('ru.png')%>", style="height: 30px;">

<%= link\_to "RU", { :locale=>'en' }, class: "local" %>

<% else %>

<img src="<%=asset\_path('eng.png')%>", style="height: 30px;">

<%= link\_to "ENG", { :locale=>'ru' } , class: "local"%>

<%end%>

</li>

<%if user\_signed\_in?%>

<li class="nav-item dropdown mr-auto">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false" id="nav-nick">

<%= session[:user\_nick]%>

</a>

<ul class="dropdown-menu dropdown-menu-end" aria-labelledby="navbarDropdown">

<li><%=button\_to t(:my\_account), "/users/#{session[:user\_id]}", class: "dropdown-item btn", method: :get, type: "button", id: "nav-myacc"%></li>

<li><%=button\_to t(:sound\_settings), "/sounds", class: "dropdown-item btn", method: :get, type: "button"%></li>

<li><hr class="dropdown-divider"></li>

<li><%= button\_to t(:logout), session\_path, method: :delete, class: "dropdown-item btn btn-lg", id: "nav-logout"%></li>

</ul>

</li>

<%else%>

<li class="nav-item">

<a class="nav-link" href="/login"><%=t(:login)%></a></i>

</li>

<li class="nav-item">

<a class="nav-link" href="/new"><%=t(:signup)%></a></i>

</li>

<%end%>

</ul>

</div>

</div>

</div>

</nav>

</header>

<div id="flash">

<% flash.each do |type, msg| %>

<%if type == "error"%>

<%type = "danger"%>

<%end%>

<div class='alert alert-<%= type%>'>

<%= msg %>

</div>

<% flash.clear %>

<% end %>

</div>

<%= yield %>

<div class="modal fade" id="new\_task" tabindex="-1" aria-labelledby="new\_task\_label" aria-hidden="true">

<div class="modal-dialog modal-xl">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="new\_task\_label"><%=t(:new\_task)%></h4>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

</div>

<div class="modal-body">

<%@task = Task.new%>

<div class= "content">

<%= render "tasks/form", task: @task %>

<br>

<div>

<%= link\_to t(:back\_to\_tasks), tasks\_path, class: 'btn btn-outline-secondary' %>

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal"><%=t(:close\_modal)%></button>

</div>

</div>

</div>

</div>

<audio id="wahwahwah" src="<%= asset\_path('wahwahwah.mp3') %>"></audio>

<audio id="sad\_violin" src="<%= asset\_path('sad\_violin.mp3') %>"></audio>

<audio id="spongebob-fail" src="<%= asset\_path('spongebob-fail.mp3') %>"></audio>

<audio id="si" src="<%= asset\_path('si.mp3') %>"></audio>

<audio id="done" src="<%= asset\_path('done.mp3') %>"></audio>

<audio id="beep" src="<%= asset\_path('beep.mp3') %>"></audio>

<audio id="coins" src="<%= asset\_path('coins.mp3') %>"></audio>

<audio id="oou" src="<%= asset\_path('oou.mp3') %>"></audio>

<audio id="piu" src="<%= asset\_path('piu.mp3') %>"></audio>

<audio id="wet" src="<%= asset\_path('wet.mp3') %>"></audio>

<audio id="accomplished" src="<%= asset\_path('accomplished.mp3') %>"></audio>

<audio id="ddd" src="<%= asset\_path('ddd.mp3') %>"></audio>

<audio id="elegant" src="<%= asset\_path('elegant.mp3') %>"></audio>

<audio id="error" src="<%= asset\_path('error.mp3') %>"></audio>

<audio id="p" src="<%= asset\_path('p.mp3') %>"></audio>

<audio id="smile-ringtone" src="<%= asset\_path('smile-ringtone.mp3') %>"></audio>

<audio id="swiftly" src="<%= asset\_path('swiftly.mp3') %>"></audio>

<audio id="system-fault" src="<%= asset\_path('system-fault.mp3') %>"></audio>

<audio id="yo" src="<%= asset\_path('yo.mp3') %>"></audio>

<footer>

<% if user\_signed\_in?%>

<div id = "never\_gonna\_give\_you\_up">

<h5><%=t(:dont\_click\_me)%></h5>

<iframe onmouseover=show() onmouseout=dont\_show() width="110" height="100" src="https://www.myinstants.com/instant/click-here/embed/" frameborder="0" scrolling="no"></iframe>

</div>

<%end%>

</footer>

</body>

<script>

function play(option) {

var audio\_good = document.getElementById(option);

audio\_good.play();

}

function show() {

var dance = document.getElementById('container');

dance.style.visibility='visible'

}

function dont\_show() {

var dance = document.getElementById('container');

dance.style.visibility='hidden'

}

</script>

</html>

**views/sessions**

**destroy.html.erb**

<div class = "main-login">

<h1><%= t(:log\_in\_please)%></h1>

<%= form\_with(url: session\_path,class: "form", id: "login-form") do |form| %>

<div>

<%= form.label t(:email), style: "display: block" %>

<%= form.text\_field :email, class: "form-control", id: "li-email", type: "email" %>

</div>

<div>

<%= form.label t(:password), style: "display: block" %>

<%= form.text\_field :password, id: "li-password",class: "form-control", type: "password" %>

</div>

<div>

<%= form.submit t(:login), class: "btn btn-primary btn-lg", style: "height: 60px;width:200px;", id: "login-btn" %>

<%= link\_to t(:signup), '/users/new', :method => "get", class: "btn btn-primary btn-lg", style: "height: 60px;width:200px;"%>

</div>

<% end %>

</div>

**views/tasks**

**\_completed\_tasks.html.erb**

<% if @tasks.where(completed: true).count.zero?%>

<p><%=t(:no\_completed\_tasks)%></p>

<%end%>

<% @tasks.all.where(completed: true).order('completed\_time DESC').reject{|task| task.completed\_time.nil?}.each do |task| %>

<% if task.completed\_time.today? && @option == "today"|| task.completed\_time.month == Date.today.month && @option == "month" || task.completed\_time.to\_date.cweek == Date.today.cweek && @option == "week"|| @option == "all"%>

<div id="<%= dom\_id task %>" class="task">

<div class = "task\_content">

<%=button\_to " ", uncomplete\_path(:task => task), class: 'btn\_instead\_of\_checkbox\_return', onclick:"play('#{current\_user.failure\_sound}')"%>

<div class = "one\_line">

<div style="text-decoration: line-through">

<p>

<%= task.title %>

</p>

</div>

<a class="link" data-bs-toggle="modal" data-bs-target="#pending<%= dom\_id task %>"><%=t(:open\_task)%></a>

</div>

</div>

<div class="modal fade" id="pending<%= dom\_id task %>" tabindex="-1" aria-labelledby="open\_label" aria-hidden="true">

<div class="modal-dialog modal-xl">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="open\_label"><%=t(:task\_label)%></h4>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

</div>

<div class="modal-body">

<%= render "task\_show", task: task %>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal"><%=t(:close\_modal)%></button>

</div>

</div>

</div>

</div>

</div>

<%end%>

<% end %>

**\_form.html.erb**

<%= form\_with(model: @task, class: 'form') do |form| %>

<% if @task.errors.any? %>

<div style="color: red">

<h2><%= pluralize(@task.errors.count, "error") %> prohibited this @task from being saved:</h2>

<ul>

<% @task.errors.each do |error| %>

<li><%= error.full\_message %></li>

<% end %>

</ul>

</div>

<% end %>

<div>

<%= form.label t(:title), style: "display: block"%>

<%= form.text\_field :title, class: 'form-control' , id: "task-title" %>

</div>

<div>

<%= form.label t(:description), style: "display: block" %>

<%= form.text\_field :description, class: 'form-control' %>

</div>

<div>

<%= form.label t(:due), style: "display: block" %>

<%= form.date\_field :due, id: "due-input", class: 'input-group date'%>

</div>

<div>

<%= form.submit t(:save\_task), class: "btn btn-success btn-lg" %>

</div>

<% end %>

**\_pending\_tasks.html.erb**

<% if @tasks.where(completed: false).count.zero?%>

<p style="font-size:20px;text-align:center"><%=t(:all\_tasks\_are\_completed)%></p>

<img src = "<%= image\_path('completed.jpg') %>" alt= "completed\_image" style="width:25%;margin-left:37%">

<%end%>

<% @tasks.all.order('created\_at DESC').each do |task| %>

<% if task.completed == false%>

<div id="<%= dom\_id task %>" class="task">

<div class = "task\_content">

<%=button\_to " ", complete\_path(task: task), class: 'btn\_instead\_of\_checkbox', onclick:"play('#{current\_user.success\_sound}')" %>

<div class = "one\_line">

<p>

<%= task.title %>

</p>

<a class="link" data-bs-toggle="modal" data-bs-target="#pending<%= dom\_id task %>"><%=t(:open\_task)%></a>

</div>

</div>

<%unless task.due.nil?%>

<div>

<p class = "due\_p" <%if task.due < Date.today()%> style="color:red"<%end%>><%if task.due < Date.today()%>

<img src="<%=asset\_path('over.png')%>" style="height:25px">

<%end%><%=t(:due)%>: <%=task.due%></p>

</div>

<%end%>

<div class="modal fade" id="pending<%= dom\_id task %>" tabindex="-1" aria-labelledby="open\_label" aria-hidden="true">

<div class="modal-dialog modal-xl">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="open\_label"><%=t(:task\_label)%></h4>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

</div>

<div class="modal-body">

<%= render "task\_show", task: task %>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal"><%=t(:close\_modal)%></button>

</div>

</div>

</div>

</div>

</div>

<%end%>

<% end %>

**\_task\_show.html.erb**

<div class = "task\_show">

<strong><%=t(:title)%>:</strong>

<%= task.title %>

<br/>

<strong><%=t(:description)%>:</strong>

<%= task.description %>

<br/>

<strong><%=t(:due)%>:</strong>

<%= task.due %>

<br/>

<% if user\_signed\_in?%>

<strong><%=t(:id)%>:</strong>

<%= task.user\_id %>

<%end%>

<br/>

<strong><%=t(:completed)%>:</strong>

<%completed = task.completed %>

<%if completed%>

<%=t(:true)%>

<%else%>

<%=t(:false)%>

<%end%>

<br/>

<div>

<%= link\_to t(:edit\_task), edit\_task\_path(task), class: 'btn btn-primary' %> <br/>

<%= button\_to t(:destroy\_task), task, method: :delete, class: 'btn btn-outline-danger'%>

</div>

</div>

**choose.turbo\_stream.erb**

<%= turbo\_stream.update 'completed-tasks', partial: 'completed\_tasks'%>

**complete.turbo\_stream.erb**

<%= turbo\_stream.update 'pending-tasks', partial: 'pending\_tasks'%>

<%= turbo\_stream.update 'completed-tasks', partial: 'completed\_tasks'%>

**edit.html.erb**

<h1><%=t(:editing\_task)%></h1>

<%= render "form", task: @task %>

<br>

<div>

<%= link\_to t(:back\_to\_tasks), tasks\_path, class: "btn btn-outline-secondary" %>

</div>

**index.html.erb**

<% if user\_signed\_in?%>

<p style="color: green"><%= notice %></p>

<%= link\_to t(:new\_task), new\_task\_path, type:"button", id: "btn-add", class: "btn glow-on-hover", style:'color:white;font-size:20px' %>

<br/>

<div id="pending-tasks">

<%= render "pending\_tasks"%>

</div>

<details >

<summary>

<%=t(:completed\_tasks)%>

<%= form\_with url: "/choose", method: :post, id: 'options' do |form|%>

<%= form.select :option, [["#{t(:today)}", "today"], ["#{t(:this\_week)}", "week"], ["#{t(:this\_month)}", "month"], ["#{t(:all)}", "all"]], {}, { class: "form form-group form-control" } %>

<%= form.submit t(:show), class: 'btn btn-outline-primary'%>

<% end %>

</summary>

<div id="completed-tasks">

<%= render "completed\_tasks"%>

</div>

</details>

<div id="page">

<div id="container" style="visibility: hidden">

<div id="foot" class="left">

<div id="f1"></div>

<div id="f2"></div>

<div id="leg1" class="first">

<div id="leg1" class="next1">

<div id="leg1" class="next1">

<div id="leg1" class="next1">

<div id="leg1" class="next1"></div>

</div>

</div>

</div>

</div>

</div>

<div id="foot" class="right">

<div id="f1"></div>

<div id="f2"></div>

<div id="leg2" class="first">

<div id="leg2" class="next2">

<div id="leg2" class="next2">

<div id="leg2" class="next2">

<div id="leg2" class="next2"></div>

</div>

</div>

</div>

</div>

</div>

<div id="body">

<div id="b1"></div>

<div id="b2"></div>

<div id="b3"></div>

<div id="head">

<div id="h1"></div>

<div id="h2"></div>

<div id="h3"></div>

<div id="h4"></div>

<div id="h5"><span></span></div>

<div id="eyes">

<div id="e">

<div id="e1">

<div id="eye"></div>

<span></span>

</div>

<div id="e2">

<div id="eye"></div>

<span></span>

</div>

</div>

</div>

<div id="mouth">

<span></span>

<span></span>

<span></span>

<span></span>

<span></span>

<div id="m1"></div>

<div id="m2"></div>

</div>

</div>

<div id="b4"></div>

<div id="b5">

<div id="b51">

<div id="b51">

<div id="b51">

<div id="b51">

<div id="ab">

<div id="a1"></div>

<div id="a2"></div>

<div id="a3" class="a31"></div>

<div id="a3" class="a32"></div>

<div id="a3" class="a33"></div>

</div>

</div>

</div>

</div>

</div>

</div>

<div id="b6"></div>

<div id="b7">

<div id="b71">

<div id="b71">

<div id="b71">

<div id="b71">

<div id="ac">

<div id="a1"></div>

<div id="a2"></div>

<div id="a3" class="a31"></div>

<div id="a3" class="a32"></div>

<div id="a3" class="a33"></div>

</div>

</div>

</div>

</div>

</div>

</div>

<div id="b8">

<span></span>

</div>

<div id="b9"></div>

</div>

</div>

</div>

<%end%>

**new.index.html.erb**

<h1><%=t(:new\_task)%></h1>

<%= render "form"%>

<div>

<%= link\_to "Back to tasks", tasks\_path, class: 'btn btn-outline-secondary' , id: 'new-task'%>

</div>

**show.html.erb**

<p style="color: green"><%= notice %></p>

<strong><%=t(:title)%>:</strong>

<%= @task.title %>

<br/>

<strong><%=t(:due)%>:</strong>

<%= @task.due %>

<% if user\_signed\_in?%>

<strong><%=t(:id)%>:</strong>

<%= @task.user\_id %>

<%end%>

<strong><%=t(:completed)%>:</strong>

<%= @task.completed %>

<div>

<%= link\_to t(:edit\_task), edit\_task\_path(@task) %> |

<%= link\_to t(:back\_to\_tasks), tasks\_path %>

<%= button\_to t(:destroy\_task) :delete %>

</div>

**uncomplete.turbo\_stream.erb**

<%= turbo\_stream.update 'pending-tasks', partial: 'pending\_tasks'%>

<%= turbo\_stream.update 'completed-tasks', partial: 'completed\_tasks'%>

**views/user\_mailer**

**registration\_confirmation.html.erb**

<h1> <%=t(:hi)%>, <%= @user.nick %>! 👋≧◉ᴥ◉≦ </h1>

<p>

<%=t(:thanks\_for\_registration)%>

<br/>

<%= url\_for confirm\_email\_user\_url(@user.confirm\_token, email: @user.email) %>

</p>

**registration\_confirmation.text.erb**

<h1> <%=t(:hi)%>, <%= @user.nick %>! 👋≧◉ᴥ◉≦ </h1>

<p>

<%=t(:thanks\_for\_registration)%>

<br/>

<%= url\_for confirm\_email\_user\_url(@user.confirm\_token, email: @user.email) %>

</p>

**views/users**

**\_form.html.erb**

<%= form\_with(model: user, class: "form") do |form| %>

<% if user.errors.any? %>

<div style="color: red">

<h2><%= pluralize(user.errors.count, "error") %> prohibited this user from being saved:</h2>

<ul>

<% user.errors.each do |error| %>

<li><%= error.full\_message %></li>

<% end %>

</ul>

</div>

<% end %>

<div>

<%= form.label t(:nick), style: "display: block" %>

<%= form.text\_field :nick, id: "su-nick", class: "form-control" %>

</div>

<div>

<%= form.label t(:name), style: "display: block" %>

<%= form.text\_field :name, id: "su-name", class: "form-control" %>

</div>

<div>

<%= form.label t(:surname), style: "display: block" %>

<%= form.text\_field :surname, id: "su-surname", class: "form-control" %>

</div>

<div>

<%= form.label t(:email), style: "display: block" %>

<%= form.text\_field :email, id: "su-email", class: "form-control", type: "email" %>

</div>

<div>

<%= form.label t(:password), style: "display: block" %>

<%= form.text\_field :password, id: "su-password", type: "password", class: "form-control" %>

</div>

<div>

<%= form.label t(:password\_confirmation), style: "display: block" %>

<%= form.text\_field :password\_confirmation, id: "su-password-conf", type: "password", class: "form-control" %>

</div>

<div>

<%= form.submit t(:signup), class: "btn btn-success btn-lg"%>

</div>

<% end %>

**\_user.htm.erb**

<div id="<%= dom\_id user %>" >

<% if user\_signed\_in? && current\_user.id == user.id%>

<h1><%=t(:my\_account)%></h1>

<div style="display:flex;">

<%= link\_to t(:edit\_my\_account), edit\_user\_path(@user), class:'btn btn-primary' %>

<%= button\_to t(:delete\_my\_account), @user, data: {turbo\_method: :delete, turbo\_confirm: 'Are you sure?'}, class: 'btn btn-danger' %>

</div>

<p>

<strong><%=t(:id)%>:</strong>

<%= user.id %>

</p>

<p>

<%end%>

<strong><%=t(:nick)%>:</strong>

<%= user.nick %>

</p>

<p>

<strong><%=t(:name)%>:</strong>

<%= user.name %>

</p>

<p>

<strong><%=t(:surname)%>:</strong>

<%= user.surname %>

</p>

<% if user\_signed\_in? && current\_user.id == user.id%>

<p>

<strong><%=t(:email)%>:</strong>

<%= user.email %>

</p>

<p>

<strong><%=t(:password)%>:</strong>

<%= user.password %>

</p>

<%end%>

<p>

<strong><%=t(:pending\_tasks)%>:</strong>

<%=@pending\_tasks.count%>

</p>

<p>

<strong><%=t(:completed\_tasks)%>:</strong>

<%=@completed\_tasks.count%>

</p>

<%if current\_user.id == user.id %>

<h1><%=t(:your\_progress)%></h1>

<%else%>

<h1><%=user.nick%><%=t(:users\_progress)%></h1>

<%end%>

<%= column\_chart @completed\_tasks.group\_by\_day(:completed\_time, last:7).count, id: "histogram"%>

</div>

**edit.html.erb**

<h1><%=t(:editing\_user)%></h1>

<%= render "form", user: @user %>

<br>

**index.html.erb**

<p style="color: green"><%= notice %></p>

<h1><%=t(:users)%></h1>

<div id="users">

<% @users.each do |user| %>

<div id="<%= dom\_id user %>" class = "user">

<p>

<strong><%=t(:nick)%>:</strong>

<%= user.nick %>

</p>

<p>

<%= link\_to t(:show\_this\_user), user, class: 'link'%>

</p>

</div>

<% end %>

</div>

**new.html.erb**

<h1><%=t(:new\_user)%></h1>

<%= form\_with(model: @user, class: "form") do |form| %>

<% if @user.errors.any? %>

<div style="color: red">

<h2><%= pluralize(@user.errors.count, "error") %> prohibited this user from being saved:</h2>

<ul>

<% @user.errors.each do |error| %>

<li><%= error.full\_message %></li>

<% end %>

</ul>

</div>

<% end %>

<div>

<%= form.label t(:nick), style: "display: block" %>

<%= form.text\_field :nick, id: "su-nick", class: "form-control" %>

</div>

<div>

<%= form.label t(:name), style: "display: block" %>

<%= form.text\_field :name, id: "su-name", class: "form-control" %>

</div>

<div>

<%= form.label t(:surname), style: "display: block" %>

<%= form.text\_field :surname, id: "su-surname", class: "form-control" %>

</div>

<div>

<%= form.label t(:email), style: "display: block" %>

<%= form.text\_field :email, id: "su-email", class: "form-control", type: "email" %>

</div>

<div>

<%= form.label t(:password), style: "display: block" %>

<%= form.text\_field :password, id: "su-password", type: "password", class: "form-control" %>

</div>

<div>

<%= form.label t(:password\_confirmation), style: "display: block" %>

<%= form.text\_field :password\_confirmation, id: "su-password-conf", type: "password", class: "form-control" %>

</div>

<div>

<%= form.submit t(:signup), class: "btn btn-success btn-lg", id: "signup-btn"%>

</div>

<% end %>

<br>

<div>

<%= link\_to "Oh, I already have an account!!!", login\_path %>

</div>

**show.html.erb**

<div class = "user-show">

<p style="color: green"><%= notice %></p>

<%if @user.id != current\_user.id%>

<div id = 'relationship\_button'>

<%if !find\_request(current\_user, @user).nil?%>

<%= render "friends/request\_is\_sent", locals: {friend2: @user.id, friend1: current\_user.id}%>

<%elsif !find\_request(@user, current\_user).nil?%>

<%= render "friends/accept\_request", locals: {friend2: @user.id, friend1: current\_user.id}%>

<%elsif !find\_friend(@user).nil? %>

<%= render "friends/destroy\_friendship", locals: {friend2: @user, friend1: current\_user}%>

<% else %>

<%= render "friends/add\_as\_friend", locals: {friend2: @user.id, friend1: current\_user.id}%>

<%end%>

</div>

<%end%>

<%= render @user %>

</div>

**sounds.html.erb**

<%= form\_with(url: '/save\_sounds', class: "form", method: :get) do |form| %>

<h1><%=t(:choose\_success\_sound)%></h1>

<div id = "succes\_radios">

<%# !!params[:success\_sound] %>

<%= form.radio\_button :success\_sound, "si" , onclick:"play('si')" %>

<%= form.label :success\_sound, "si", value: "si" %><br />

<%= form.radio\_button :success\_sound, "coins", onclick:"play('coins')" %>

<%= form.label :success\_sound, "coins", value: "coins" %><br />

<%= form.radio\_button :success\_sound, "accomplished", onclick:"play('accomplished')" %>

<%= form.label :success\_sound, "accomplished", value: "accomplished" %><br />

<%= form.radio\_button :success\_sound, "beep", onclick:"play('beep')" %>

<%= form.label :success\_sound, "beep", value: "beep" %><br />

<%= form.radio\_button :success\_sound, "done", onclick:"play('done')" %>

<%= form.label :success\_sound, "done", value: "done" %><br />

<%= form.radio\_button :success\_sound, "elegant", onclick:"play('elegant')" %>

<%= form.label :success\_sound, "elegant", value: "elegant" %><br />

<%= form.radio\_button :success\_sound, "smile-ringtone", onclick:"play('smile-ringtone')" %>

<%= form.label :success\_sound, "smile-ringtone", value: "smile-ringtone" %><br />

<%= form.radio\_button :success\_sound, "swiftly", onclick:"play('swiftly')" %>

<%= form.label :success\_sound, "swiftly", value: "swiftly" %><br />

<%= form.radio\_button :success\_sound, "wet", onclick:"play('wet')" %>

<%= form.label :success\_sound, "wet", value: "wet" %><br />

<%= form.radio\_button :success\_sound, "yo", onclick:"play('yo')" %>

<%= form.label :success\_sound, "yo", value: "you" %><br />

</div>

<h1><%=t(:choose\_failure\_sound)%></h1>

<div id = "failure\_radios">

<%# !!params[:success\_sound] %>

<%= form.radio\_button :failure\_sound, "ddd" , onclick:"play('ddd')"%>

<%= form.label :failure\_sound, "ddd", value: "ddd"%><br />

<%= form.radio\_button :failure\_sound, "error", onclick:"play('error')" %>

<%= form.label :failure\_sound, "error", value: "error" %><br />

<%= form.radio\_button :failure\_sound, "oou", onclick:"play('oou')" %>

<%= form.label :failure\_sound, "oou", value: "oou" %><br />

<%= form.radio\_button :failure\_sound, "p", onclick:"play('p')" %>

<%= form.label :failure\_sound, "p", value: "p" %><br />

<%= form.radio\_button :failure\_sound, "piu", onclick:"play('piu')" %>

<%= form.label :failure\_sound, "piu", value: "piu" %><br />

<%= form.radio\_button :failure\_sound, "sad\_violin", onclick:"play('sad\_violin')" %>

<%= form.label :failure\_sound, "sad\_violin", value: "sad\_violin" %><br />

<%= form.radio\_button :failure\_sound, "spongebob-fail", onclick:"play('spongebob-fail')" %>

<%= form.label :failure\_sound, "spongebob-fail", value: "spongebob-fail" %><br />

<%= form.radio\_button :failure\_sound, "system-fault", onclick:"play('system-fault')" %>

<%= form.label :failure\_sound, "system-fault", value: "system-fault" %><br />

<%= form.radio\_button :failure\_sound, "wahwahwah", onclick:"play('wahwahwah')" %>

<%= form.label :failure\_sound, "wahwahwah", value: "wahwahwah" %><br />

</div>

<div>

<%= form.submit t(:save\_sounds), class: "btn btn-success btn-lg"%>

</div>

<% end %>

**config**

**environments/development.rb**

# frozen\_string\_literal: true

require 'active\_support/core\_ext/integer/time'

Rails.application.configure do

config.cache\_classes = false

# Do not eager load code on boot.

config.eager\_load = false

# Show full error reports.

config.consider\_all\_requests\_local = true

# Enable server timing

config.server\_timing = true

# Enable/disable caching. By default caching is disabled.

# Run rails dev:cache to toggle caching.

if Rails.root.join('tmp/caching-dev.txt').exist?

config.action\_controller.perform\_caching = true

config.action\_controller.enable\_fragment\_cache\_logging = true

config.cache\_store = :memory\_store

config.public\_file\_server.headers = {

'Cache-Control' => "public, max-age=#{2.days.to\_i}"

}

else

config.action\_controller.perform\_caching = false

config.cache\_store = :null\_store

end

# Store uploaded files on the local file system (see config/storage.yml for options).

config.active\_storage.service = :local

config.action\_mailer.perform\_caching = false

config.action\_mailer.perform\_deliveries = true

config.action\_mailer.raise\_delivery\_errors = true

config.action\_mailer.delivery\_method = :smtp

config.action\_mailer.default\_url\_options = { host: 'localhost:3000', protocol: 'http' }

config.action\_mailer.smtp\_settings = {

address: 'smtp.gmail.com',

port: 465,

user\_name: ENV['TODO\_APP\_GMAIL'],

password: ENV['TODO\_APP\_PASSWORD'],

authentication: 'plain',

ssl: true

}

# Print deprecation notices to the Rails logger.

config.active\_support.deprecation = :log

# Raise exceptions for disallowed deprecations.

config.active\_support.disallowed\_deprecation = :raise

# Tell Active Support which deprecation messages to disallow.

config.active\_support.disallowed\_deprecation\_warnings = []

# Raise an error on page load if there are pending migrations.

config.active\_record.migration\_error = :page\_load

# Highlight code that triggered database queries in logs.

config.active\_record.verbose\_query\_logs = true

# Suppress logger output for asset requests.

config.assets.quiet = true

end

**environments/production.rb**

# frozen\_string\_literal: true

require 'active\_support/core\_ext/integer/time'

Rails.application.configure do

# Settings specified here will take precedence over those in config/application.rb.

# Code is not reloaded between requests.

config.cache\_classes = true

# Eager load code on boot. This eager loads most of Rails and

# your application in memory, allowing both threaded web servers

# and those relying on copy on write to perform better.

# Rake tasks automatically ignore this option for performance.

config.eager\_load = true

# Full error reports are disabled and caching is turned on.

config.consider\_all\_requests\_local = false

config.action\_controller.perform\_caching = true

# Ensures that a master key has been made available in either ENV["RAILS\_MASTER\_KEY"]

# or in config/master.key. This key is used to decrypt credentials (and other encrypted files).

# config.require\_master\_key = true

config.action\_mailer.default\_url\_options = { host: 'localhost', port: 3000 }

# Disable serving static files from the `/public` folder by default since

# Apache or NGINX already handles this.

config.public\_file\_server.enabled = ENV['RAILS\_SERVE\_STATIC\_FILES'].present?

# Compress CSS using a preprocessor.

# config.assets.css\_compressor = :sass

# Do not fallback to assets pipeline if a precompiled asset is missed.

config.assets.compile = false

# Enable serving of images, stylesheets, and JavaScripts from an asset server.

# config.asset\_host = "http://assets.example.com"

# Specifies the header that your server uses for sending files.

# config.action\_dispatch.x\_sendfile\_header = "X-Sendfile" # for Apache

# config.action\_dispatch.x\_sendfile\_header = "X-Accel-Redirect" # for NGINX

# Store uploaded files on the local file system (see config/storage.yml for options).

config.active\_storage.service = :local

# Mount Action Cable outside main process or domain.

# config.action\_cable.mount\_path = nil

# config.action\_cable.url = "wss://example.com/cable"

# config.action\_cable.allowed\_request\_origins = [ "http://example.com", /http:\/\/example.\*/ ]

# Force all access to the app over SSL, use Strict-Transport-Security, and use secure cookies.

# config.force\_ssl = true

# Include generic and useful information about system operation, but avoid logging too much

# information to avoid inadvertent exposure of personally identifiable information (PII).

config.log\_level = :info

# Prepend all log lines with the following tags.

config.log\_tags = [:request\_id]

# Use a different cache store in production.

# config.cache\_store = :mem\_cache\_store

# Use a real queuing backend for Active Job (and separate queues per environment).

# config.active\_job.queue\_adapter = :resque

# config.active\_job.queue\_name\_prefix = "wishlist\_production"

config.action\_mailer.perform\_caching = false

# Ignore bad email addresses and do not raise email delivery errors.

# Set this to true and configure the email server for immediate delivery to raise delivery errors.

# config.action\_mailer.raise\_delivery\_errors = false

# Enable locale fallbacks for I18n (makes lookups for any locale fall back to

# the I18n.default\_locale when a translation cannot be found).

config.i18n.fallbacks = true

# Don't log any deprecations.

config.active\_support.report\_deprecations = false

# Use default logging formatter so that PID and timestamp are not suppressed.

config.log\_formatter = ::Logger::Formatter.new

# Use a different logger for distributed setups.

# require "syslog/logger"

# config.logger = ActiveSupport::TaggedLogging.new(Syslog::Logger.new "app-name")

if ENV['RAILS\_LOG\_TO\_STDOUT'].present?

logger = ActiveSupport::Logger.new($stdout)

logger.formatter = config.log\_formatter

config.logger = ActiveSupport::TaggedLogging.new(logger)

end

# Do not dump schema after migrations.

config.active\_record.dump\_schema\_after\_migration = false

end

**environments/test.rb**

# frozen\_string\_literal: true

require 'active\_support/core\_ext/integer/time'

# The test environment is used exclusively to run your application's

# test suite. You never need to work with it otherwise. Remember that

# your test database is "scratch space" for the test suite and is wiped

# and recreated between test runs. Don't rely on the data there!

Rails.application.configure do

# Settings specified here will take precedence over those in config/application.rb.

# Turn false under Spring and add config.action\_view.cache\_template\_loading = true.

config.cache\_classes = true

# Eager loading loads your whole application. When running a single test locally,

# this probably isn't necessary. It's a good idea to do in a continuous integration

# system, or in some way before deploying your code.

config.eager\_load = ENV['CI'].present?

# Configure public file server for tests with Cache-Control for performance.

config.public\_file\_server.enabled = true

config.public\_file\_server.headers = {

'Cache-Control' => "public, max-age=#{1.hour.to\_i}"

}

# Show full error reports and disable caching.

config.consider\_all\_requests\_local = true

config.action\_controller.perform\_caching = false

config.cache\_store = :null\_store

# Raise exceptions instead of rendering exception templates.

config.action\_dispatch.show\_exceptions = false

# Disable request forgery protection in test environment.

config.action\_controller.allow\_forgery\_protection = false

# Store uploaded files on the local file system in a temporary directory.

config.active\_storage.service = :test

config.action\_mailer.perform\_caching = false

# Tell Action Mailer not to deliver emails to the real world.

# The :test delivery method accumulates sent emails in the

# ActionMailer::Base.deliveries array.

config.action\_mailer.delivery\_method = :test

# Print deprecation notices to the stderr.

config.active\_support.deprecation = :stderr

# Raise exceptions for disallowed deprecations.

config.active\_support.disallowed\_deprecation = :raise

# Tell Active Support which deprecation messages to disallow.

config.active\_support.disallowed\_deprecation\_warnings = []

config.action\_mailer.delivery\_method = :smtp

config.action\_mailer.default\_url\_options = { host: 'localhost:3000', protocol: 'http' }

config.action\_mailer.smtp\_settings = {

address: 'smtp.gmail.com',

port: 465,

user\_name: ENV['TODO\_APP\_GMAIL'],

password: ENV['TODO\_APP\_PASSWORD'],

authentication: 'plain',

ssl: true

}

# Raises error for missing translations.

# config.i18n.raise\_on\_missing\_translations = true

# Annotate rendered view with file names.

# config.action\_view.annotate\_rendered\_view\_with\_filenames = true

end

**routes**

# frozen\_string\_literal: true

Rails.application.routes.draw do

scope '(:locale)', locale: /#{I18n.available\_locales.join('|')}/ do

resources :tasks

resource :session, only: %i[new create destroy]

resources :sessions, only: %i[new create show]

resources :users

root 'tasks#index'

get 'login', to: 'sessions#new'

get 'new', to: 'users#new'

get '/users/:user\_id', to: 'users#show'

post 'choose\_color', to: 'users#choose\_color'

get 'sounds', to: 'users#sounds'

get 'save\_sounds', to: 'users#save\_sounds'

get 'month', to: 'calendars#month'

get 'week', to: 'calendars#week'

post '/complete', to: 'tasks#complete'

post '/uncomplete', to: 'tasks#uncomplete'

post '/choose', to: 'tasks#choose'

post 'send\_request', to: 'friends#send\_request'

post 'cancell\_request', to: 'friends#cancell\_request'

post 'accept\_request', to: 'friends#accept\_request'

post 'decline\_request', to: 'friends#decline\_request'

post 'destroy\_friendship', to: 'friends#destroy\_friendship'

get 'friends', to: 'friends#index'

get 'friends\_competitions', to: 'competitions#friends'

get 'all\_competitions', to: 'competitions#all\_users'

resources :users do

member do

get :confirm\_email

# get :registration\_confirmation

end

end

end

end

**database.yml**

# SQLite. Versions 3.8.0 and up are supported.

# gem install sqlite3

#

# Ensure the SQLite 3 gem is defined in your Gemfile

# gem "sqlite3"

#

default: &default

adapter: postgresql

encoding: unicode

user: liubasik

password: <%= ENV["MERSENNE\_DATABASE\_PASSWORD"] %>

pool: <%= ENV.fetch("RAILS\_MAX\_THREADS") { 5 } %>

development:

<<: \*default

database: todolist\_development

# Warning: The database defined as "test" will be erased and

# re-generated from your development database when you run "rake".

# Do not set this db to the same as development or production.

test:

<<: \*default

database: todolist\_test

production:

<<: \*default

database: todolist\_production

**Файлы локализации:**

**en.yml**

# Files in the config/locales directory are used for internationalization

# and are automatically loaded by Rails. If you want to use locales other

# than English, add the necessary files in this directory.

#

# To use the locales, use `I18n.t`:

#

# I18n.t "hello"

#

# In views, this is aliased to just `t`:

#

# <%= t("hello") %>

#

# To use a different locale, set it with `I18n.locale`:

#

# I18n.locale = :es

#

# This would use the information in config/locales/es.yml.

#

# The following keys must be escaped otherwise they will not be retrieved by

# the default I18n backend:

#

# true, false, on, off, yes, no

#

# Instead, surround them with single quotes.

#

# en:

# "true": "foo"

#

# To learn more, please read the Rails Internationalization guide

# available at https://guides.rubyonrails.org/i18n.html.

en:

week\_calendar: "Week Calendar"

month\_calendar: "Month Calendar"

open\_task: "Open"

task\_label: "Task"

close\_modal: "Close"

all\_users\_competitions: "Here you can see other users' progress and compare with yours"

compete\_with\_friends\_button: "Show friends only"

all\_friends\_competitions: "Here you can see your friends' progress and compare with yours"

compete\_with\_all\_users\_button: "Show all users"

accept\_friendship\_button: "Accept friendship"

add\_as\_friend\_button: "Add as friend"

decline\_request\_button: "Decline request"

destroy\_friendship: "Destroy friendship"

nick: "Nickname"

show\_this\_friend: "Show this friend"

request\_is\_sent\_button: "Request is sent"

new\_friend\_requests: "New friend requests!"

friends: "Friends"

find\_new\_friends: "Find new friends"

make\_friends: "Friends make out lives better! Add some friends!"

todo: "ToDo"

calendar: "Calendar"

my\_friends: "My friends"

competitions: "Competitions"

my\_account: "My account"

sound\_settings: "Sound settings"

logout: "Log Out"

login: "Log In"

signup: "Sign Up"

new\_task: "New task"

back\_to\_tasks: "Back to tasks"

dont\_click\_me: "Don't click on me!"

log\_in\_please: "Log in please"

email: "Email"

password: "Password"

no\_completed\_tasks: "You have no completed tasks yet."

title: "Title"

description: "Description"

due: "Due date"

id: "User ID"

completed: "Completed"

save\_task: "Save Task"

true: "Yes"

false: "No"

all\_tasks\_are\_completed: "All tasks are completed! Congratulations"

edit\_task: "Edit task"

destroy\_task: "Delete this task"

editing\_task: "Editing task"

completed\_tasks: "Completed tasks"

today: "today"

this\_week: "this week"

this\_month: "this\_month"

all: "all the time"

show: "show"

hi: "Hi"

thanks\_for\_registration: "Thanks for signing up! For continuing follow the link below"

confirm: "Confirm"

name: "Name"

surname: "Surname"

password\_confirmation: "Password confirmation"

edit\_my\_account: "Edit my account"

editing\_user: "Editing your account info"

delete\_my\_account: "Delete my account"

pending\_tasks: "Pending tasks"

your\_progress: "Your progress this week"

users\_progress: "'s progress this week"

users: "Users"

show\_this\_user: "Show this user"

choose\_success\_sound: "Choose sound for task accomplishment"

choose\_failure\_sound: "Choose sound for making task uncompleted"

save\_sounds: "Save sounds"

already\_exists: "already exists"

cannot\_be\_blank: "cannot be blank"

you\_are\_not\_logged\_in: "You are not logged in"

you\_are\_logged\_in: "You are already logged in"

you\_have\_successfully\_logged\_in: "You have successfully logged in"

confirm\_you\_email: "Confirm your email"

incorrect\_password\_or\_email: "Incorrect password or email"

successfull\_exit: "Successfull exit"

task\_was\_created: "Task was created"

task\_was\_updated: "Task was updated"

task\_was\_destroyed: "Task was destroyed"

email\_sent: "Confirmation message has been sent to your email.Please, check it and come back again"

user\_was\_updated: "User was updated"

user\_was\_destroyed: "User was destroyed"

new\_user: "New User"

**ru.yml**

ru:

week\_calendar: "Неделя"

month\_calendar: "Месяц"

open\_task: "Открыть"

task\_label: "Задача"

close\_modal: "Закрыть"

all\_users\_competitions: "Здесь вы можете сравнить свой прогресс с другими пользователями"

compete\_with\_friends\_button: "Показать только моих друзей"

all\_friends\_competitions: "Здесь вы можете сравнить свой прогресс с друзьями"

compete\_with\_all\_friends\_button: "Показать всех пользователей"

accept\_friendship\_button: "Принять заявку"

add\_as\_friend\_button: "Добавить в друзья"

decline\_request\_button: "Отклонить заявку"

destroy\_friendship: "Удалить от друзей"

nick: "Кликуха"

show\_this\_friend: "Показать профиль друга"

request\_is\_sent\_button: "Запрос отправлен"

new\_friend\_requests: "Новые заявки в друзья!"

friends: "Друзья"

find\_new\_friends: "Найти новых друзей"

make\_friends: "Друзья делают нашу жизнь ярче! Добавьте друзей!"

todo: "ТуДуМц"

calendar: "Календарь"

my\_friends: "Мои друзья"

competitions: "Соревновалки"

my\_account: "Мой профиль"

sound\_settings: "Настройки звуков"

logout: "Выход"

login: "Вход"

signup: "Регистрация"

new\_task: "Новая задача"

back\_to\_tasks: "Вернуться к задачам"

dont\_click\_me: "Не нажимай на меня!"

log\_in\_please: "Войдите, пожалуйста"

email: "Почта"

password: "Пароль"

no\_completed\_tasks: "У вас пока нет выполненных задач"

title: "Название"

description: "Описание"

due: "Крайний срок"

id: "ID пользователя"

completed: "Выполнено"

save\_task: "Добавить задачу"

true: "Да"

false: "Нет"

all\_tasks\_are\_completed: "Все задачи выполнены! Поздравляем!"

edit\_task: "Редактировать задачу"

destroy\_task: "Удалить задачу"

editing\_task: "Редактирование задачи"

completed\_tasks: "Выполненные задачи"

today: "сегодня"

this\_week: "на этой неделе"

this\_month: "в этом месяце"

all: "за все время"

show: "показать"

hi: "Привет"

thanks\_for\_registration: "Спасибо за регистрацию. Для продолжения перейдите по ссылке"

confirm: "Подтвердить"

name: "Имя"

surname: "Фамилия"

password\_confirmation: "Подтверждение пароля"

edit\_my\_account: "Редактировать мой профиль"

editing\_user: "Редактирование профиля"

delete\_my\_account: "Удалить мой аккаунт"

pending\_tasks: "Текущие задачи"

your\_progress: "Ваш прогресс за эту неделю"

users\_progress: " сделал такой прогресс за неделю"

users: "Пользователи"

show\_this\_user: "Показать профиль этого пользователя"

choose\_success\_sound: "Выберите звук для выполнения задачи"

choose\_failure\_sound: "Выберите звук для отмены выполнения задачи"

save\_sounds: "Сохранить звуки"

already\_exists: "уже существует"

cannot\_be\_blank: "не может быть пустым"

you\_are\_not\_logged\_in: "Вы не авторизованы"

you\_are\_logged\_in: "Вы уже авторизованы"

you\_have\_successfully\_logged\_in: "Вы успешно вошли в аккаунт"

confirm\_you\_email: "Подтвердите почту"

incorrect\_password\_or\_email: "Неправильно указаны пароль или почта"

successfull\_exit: "Успешный выход"

task\_was\_created: "Задача была создана"

task\_was\_updated: "Задача была обновлена"

task\_was\_destroyed: "Задача была удалена"

email\_sent: "На указанную почту выслано письмо. Подтвердите почту, пожалуйста."

user\_was\_updated: "Профиль был обновлен"

user\_was\_destroyed: "Аккаунт был удален"

new\_user: "Регистрация"

**db/schema.rb**

# frozen\_string\_literal: true

# This file is auto-generated from the current state of the database. Instead

# of editing this file, please use the migrations feature of Active Record to

# incrementally modify your database, and then regenerate this schema definition.

#

# This file is the source Rails uses to define your schema when running `bin/rails

# db:schema:load`. When creating a new database, `bin/rails db:schema:load` tends to

# be faster and is potentially less error prone than running all of your

# migrations from scratch. Old migrations may fail to apply correctly if those

# migrations use external dependencies or application code.

#

# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema[7.0].define(version: 20\_221\_226\_112\_056) do

# These are extensions that must be enabled in order to support this database

enable\_extension 'plpgsql'

create\_table 'friends', force: :cascade do |t|

t.integer 'friend1', null: false

t.integer 'friend2', null: false

t.string 'relationship'

t.integer 'who\_sent\_request'

t.datetime 'created\_at', null: false

t.datetime 'updated\_at', null: false

end

create\_table 'tasks', force: :cascade do |t|

t.string 'title'

t.string 'description'

t.date 'due'

t.boolean 'reminder'

t.time 'reminder\_time'

t.boolean 'completed', default: false

t.datetime 'created\_at', null: false

t.datetime 'updated\_at', null: false

t.bigint 'user\_id'

t.datetime 'completed\_time'

t.index ['user\_id'], name: 'index\_tasks\_on\_user\_id'

end

create\_table 'users', force: :cascade do |t|

t.string 'nick', null: false

t.string 'name'

t.string 'surname'

t.string 'email', null: false

t.string 'password\_digest', null: false

t.datetime 'created\_at', null: false

t.datetime 'updated\_at', null: false

t.boolean 'email\_confirmed', default: false

t.string 'confirm\_token'

t.string 'color'

t.string 'success\_sound', default: 'si'

t.string 'failure\_sound', default: 'wahwahwah'

end

end

**Файлы стилей, миграций и т. д. Можно посмотреть в прикрепленном архиве.**

**Тесты.**

**Features**

**application\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'tasks', type: :feature do

scenario 'not autorized to root\_path' do

visit root\_path

expect(page).to have\_text('Log in please')

end

scenario 'not autorized to tasks\_path' do

visit tasks\_path

expect(page).to have\_text('Log in please')

end

scenario 'not autorized to all competitions' do

visit all\_competitions\_path

expect(page).to have\_text('Log in please')

end

scenario 'not autorized to friends competitions' do

visit friends\_competitions\_path

expect(page).to have\_text('Log in please')

end

# scenario 'not autorized to friends' do

# visit friends\_path

# expect(page).to have\_text('Log in please')

# end

# scenario 'not autorized to my account' do

# visit edit\_user\_account

# expect(page).to have\_text('Log in please')

# end

scenario 'not autorized to friends' do

visit 'tasks/new'

expect(page).to have\_text('Log in please')

end

end

**helpers**

**friends\_helper\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

# Specs in this file have access to a helper object that includes

# the FriendsHelper. For example:

#

# describe FriendsHelper do

# describe "string concat" do

# it "concats two strings with spaces" do

# expect(helper.concat\_strings("this","that")).to eq("this that")

# end

# end

# end

RSpec.describe FriendsHelper, type: :helper do

describe 'requests' do

context 'check request' do

let!(:friend1) { User.create({ nick: 'qwe', email: 'asd@asd', password: 'qwe', password\_confirmation: 'qwe' }) }

let!(:friend2) { User.create({ nick: 'asd', email: 'qwe@asd', password: 'qwe', password\_confirmation: 'qwe' }) }

let!(:request) do

Friend.create({ friend1: friend1.id, friend2: friend2.id, relationship: 'request',

who\_sent\_request: friend1.id })

end

it 'friend1 must be valid' do

expect(friend1.valid?).to eq(true)

end

it 'friend2 must be valid' do

expect(friend2.valid?).to eq(true)

end

it 'request must be valid' do

expect(request.valid?).to eq(true)

end

end

context 'when requests are sent' do

let!(:friend1) { User.create({ nick: 'qwe', email: 'asd@asd', password: 'qwe', password\_confirmation: 'qwe' }) }

let!(:friend2) { User.create({ nick: 'asd', email: 'qwe@asd', password: 'qwe', password\_confirmation: 'qwe' }) }

let!(:request) do

Friend.create({ friend1: friend1.id, friend2: friend2.id, relationship: 'request',

who\_sent\_request: friend1.id })

end

it 'must find the request' do

request = find\_request(friend1, friend2)

expect(request.relationship).to eq('request')

end

it 'must write who sent request' do

request = find\_request(friend1, friend2)

expect(request.who\_sent\_request).to eq(friend1.id)

end

it 'must write second friend' do

request = find\_request(friend1, friend2)

expect(request.friend2).to eq(friend2.id)

end

end

context 'make friendship' do

let!(:current\_user) do

User.create({ nick: 'qwe', email: 'asd@asd', password: 'qwe', password\_confirmation: 'qwe' })

end

let!(:friend2) { User.create({ nick: 'asd', email: 'qwe@asd', password: 'qwe', password\_confirmation: 'qwe' }) }

let!(:request) do

Friend.create({ friend1: current\_user.id, friend2: friend2.id, relationship: 'request',

who\_sent\_request: current\_user.id })

end

it 'will change request to friendship' do

request = find\_request(current\_user, friend2)

request.update\_attribute(:relationship, 'friendship')

expect(request.relationship).to eq('friendship')

end

it 'finds friendship' do

request = find\_request(current\_user, friend2)

request.update\_attribute(:relationship, 'friendship')

request = find\_friend(friend2)

expect(request.relationship).to eq('friendship')

end

it 'finds friend id' do

request = find\_request(current\_user, friend2)

request.update\_attribute(:relationship, 'friendship')

request = find\_friend(friend2)

expect(request.friend2).to eq(friend2.id)

end

end

end

end

**models**

**friend\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe Friend, type: :model do

describe 'validations of presence' do

context 'friend 1' do

it { should validate\_presence\_of(:friend1).with\_message(I18n.t(:cannot\_be\_blank)) }

end

context 'friend 2' do

it { should validate\_presence\_of(:friend2).with\_message(I18n.t(:cannot\_be\_blank)) }

end

end

describe 'requests' do

let!(:friends) do

{ friend1: 10, friend2: 15, relationship: 'request', who\_sent\_request: 10 }

end

let!(:friendship) { Friend.create(friends) }

context 'friend1 sends request to freidn2' do

it 'expects relationship to equal request' do

expect(friendship.relationship).to eq('request')

end

it 'expects friend 1 be the one who sent the request' do

expect(friendship.who\_sent\_request).to eq(friends[:friend1])

end

end

context 'friend 2 accepts friendship' do

it 'expects relationship to equal request' do

expect(friendship.relationship).to eq('request')

end

it 'expects relationship to be frienship' do

friendship.update\_attribute(:relationship, 'friendship')

expect(friendship.relationship).to eq('friendship')

end

end

end

end

**task\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe Task, type: :model do

describe 'validations of presence' do

context 'title' do

it { should validate\_presence\_of(:title).with\_message(I18n.t(:cannot\_be\_blank)) }

end

context 'user\_id' do

it 'will be falsy' do

task = Task.create({ title: 'task', user\_id: 'nil' })

expect(task.valid?).to eq(false)

end

end

end

describe 'changes' do

let!(:task) { Task.create({ title: 'Task for tests' }) }

context 'without changes' do

it 'will be the same' do

expect(task.title).to eq('Task for tests')

end

end

context 'with changes' do

it 'will change title' do

task.update\_attribute(:title, 'another title')

expect(task.title).to eq('another title')

end

end

end

end

**user\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe User, type: :model do

describe 'validations of presence' do

context 'nick' do

it { should validate\_presence\_of(:nick).with\_message(I18n.t(:cannot\_be\_blank)) }

end

context 'email' do

it { should validate\_presence\_of(:nick).with\_message(I18n.t(:cannot\_be\_blank)) }

end

context 'password' do

it { should validate\_presence\_of(:password).with\_message(I18n.t(:cannot\_be\_blank)) }

end

context 'password confirmation' do

it { should validate\_presence\_of(:password\_confirmation).with\_message(I18n.t(:cannot\_be\_blank)) }

end

end

describe 'validations of uniquenes' do

let!(:unique) do

{ nick: 'aaa', email: 'test@aaa', password: '1234', password\_confirmation: '1234' }

end

let!(:nick\_repeated) do

{ nick: 'aaa', email: 'test2@test', password: '1234', password\_confirmation: '1234' }

end

let!(:email\_repeated) do

{ nick: 'test2', email: 'test@aaa', password: '1234', password\_confirmation: '1234' }

end

let!(:first\_user) { User.create(unique) }

let!(:second\_user) { User.new(nick\_repeated) }

let!(:third\_user) { User.new(email\_repeated) }

it 'creates first user' do

expect(first\_user.valid?).to eq(true)

end

it 'does not create second user' do

expect(second\_user.valid?).to eq(false)

end

it 'does not create third user' do

expect(third\_user.valid?).to eq(false)

end

end

describe 'validations of correctness' do

let!(:user\_with\_wrong\_email) { User.create(email: 'wrongemail') }

it 'does not create this user' do

expect(user\_with\_wrong\_email.valid?).to eq(false)

end

end

end

**requests**

**competitions\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'Competitions', type: :request do

describe 'GET /friends' do

it 'returns http success' do

get '/friends\_competitions'

expect(response).to have\_http\_status(:found)

end

end

describe 'GET /all\_users' do

it 'returns http success' do

get '/all\_competitions'

expect(response).to have\_http\_status(:found)

end

end

end

**routing**

**task\_routing\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe TasksController, type: :routing do

describe 'routing' do

it 'routes to #index' do

expect(get: '/tasks').to route\_to('tasks#index')

end

it 'routes to #new' do

expect(get: '/tasks/new').to route\_to('tasks#new')

end

it 'routes to #show' do

expect(get: '/tasks/1').to route\_to('tasks#show', id: '1')

end

it 'routes to #edit' do

expect(get: '/tasks/1/edit').to route\_to('tasks#edit', id: '1')

end

it 'routes to #create' do

expect(post: '/tasks').to route\_to('tasks#create')

end

it 'routes to #update via PUT' do

expect(put: '/tasks/1').to route\_to('tasks#update', id: '1')

end

it 'routes to #update via PATCH' do

expect(patch: '/tasks/1').to route\_to('tasks#update', id: '1')

end

it 'routes to #destroy' do

expect(delete: '/tasks/1').to route\_to('tasks#destroy', id: '1')

end

end

end

**users\_routing\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe UsersController, type: :routing do

describe 'routing' do

it 'routes to #index' do

expect(get: '/users').to route\_to('users#index')

end

it 'routes to #new' do

expect(get: '/users/new').to route\_to('users#new')

end

it 'routes to #show' do

expect(get: '/users/1').to route\_to('users#show', id: '1')

end

it 'routes to #edit' do

expect(get: '/users/1/edit').to route\_to('users#edit', id: '1')

end

it 'routes to #create' do

expect(post: '/users').to route\_to('users#create')

end

it 'routes to #update via PUT' do

expect(put: '/users/1').to route\_to('users#update', id: '1')

end

it 'routes to #update via PATCH' do

expect(patch: '/users/1').to route\_to('users#update', id: '1')

end

it 'routes to #destroy' do

expect(delete: '/users/1').to route\_to('users#destroy', id: '1')

end

end

end

**system**

**calendars\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'calendars', type: :system do

describe 'calendars' do

scenario 'will open month page' do

user = User.create(nick: 'another', email: 'another@testr', password: '1234',

password\_confirmation: '1234', email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/month'

expect(page).to have\_text(I18n.t(:month\_calendar))

end

scenario 'will open week page' do

user = User.create(nick: 'another', email: 'another@testr', password: '1234',

password\_confirmation: '1234', email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/week'

expect(page).to have\_text(I18n.t(:week\_calendar))

end

scenario 'will open week page' do

user = User.create(nick: 'another', email: 'another@testr', password: '1234',

password\_confirmation: '1234', email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/month'

find('#month-calendar').click

sleep(1)

expect(page).to have\_text(I18n.t(:week\_calendar))

end

end

end

**competitions\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'tasks', type: :system do

describe 'login' do

scenario 'will open friends competitions' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/friends\_competitions'

expect(page).to have\_text(I18n.t(:compete\_with\_all\_users\_button))

end

scenario 'will open all users competitions' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/all\_competitions'

expect(page).to have\_text(I18n.t(:compete\_with\_friends\_button))

end

# scenario 'will open friends after all users' do

# user = User.create(nick: "new", email: "new@testr", password: "1234", password\_confirmation: "1234", email\_confirmed: true)

# visit '/login'

# fill\_in 'li-email', with: user.email

# fill\_in 'li-password', with: user.password

# find('#login-btn').click

# sleep(1)

# visit '/all\_competitions'

# sleep(1)

# find('#compete\_with\_friends')

# sleep(1)

# expect(page).to have\_text(I18n.t(:compete\_with\_all\_users\_button))

# end

end

end

**friends\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'tasks', type: :system do

describe 'tasks' do

scenario 'will open friends page' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/friends'

expect(page).to have\_text(I18n.t(:find\_new\_friends))

end

scenario 'will open friends page' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

sleep(1)

visit '/users'

expect(page).to have\_text(I18n.t(:users))

end

end

end

**tasks\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'tasks', type: :system do

describe 'login' do

scenario 'will see new task button' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

expect(page).to have\_text(I18n.t(:new\_task))

end

scenario 'will see no tasks' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

expect(page).to have\_text(I18n.t(:all\_tasks\_are\_completed))

end

scenario 'will click on new task button' do

user = User.create(nick: 'new', email: 'new@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

visit '/login'

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

find('#btn-add').click

expect(page).to have\_text(I18n.t(:due))

end

# scenario 'will click on new task button' do

# user = User.create(nick: "new", email: "new@testr", password: "1234", password\_confirmation: "1234", email\_confirmed: true)

# visit '/login'

# fill\_in 'li-email', with: user.email

# fill\_in 'li-password', with: user.password

# find('#login-btn').click

# find('#btn-add').click

# expect(page).to have\_text(I18n.t(:due))

# end

end

end

**users\_spec.rb**

# frozen\_string\_literal: true

require 'rails\_helper'

RSpec.describe 'users', type: :system do

describe 'creating user' do

scenario 'will create successfully' do

visit new\_path

fill\_in 'su-nick', with: 'tttt'

fill\_in 'su-name', with: 'test'

fill\_in 'su-email', with: 'ttt@ttt'

fill\_in 'su-password', with: '1234'

fill\_in 'su-password-conf', with: '1234'

find('#signup-btn').click

expect(page).to have\_text(I18n.t(:email\_sent))

end

end

describe 'login' do

user = User.create(nick: 'test\_userr', email: 'test@testr', password: '1234', password\_confirmation: '1234',

email\_confirmed: true)

scenario 'will login successfully' do

visit login\_path

fill\_in 'li-email', with: user.email

fill\_in 'li-password', with: user.password

find('#login-btn').click

expect(page).to have\_text(I18n.t(:you\_have\_successfully\_logged\_in))

# visit 'users/1/edit'

# fill\_in 'email', with: user.email

# find\_field('passw').set('98765')

# find\_field('passw\_conf').set('98765')

# fill\_in 'number', with: '+79121231212'

# find\_field('passw\_old').set(user.password)

# find('#btn\_up\_user').click

# expect(page).to have\_text('Профиль обновлен')

end

end

# describe 'other things' do

# scenario 'will open account' do

# user = User.create(nick: "another", email: "another@testr", password: "1234", password\_confirmation: "1234", email\_confirmed: true)

# visit '/login'

# fill\_in 'li-email', with: user.email

# fill\_in 'li-password', with: user.password

# find('#login-btn').click

# sleep(1)

# find('#nav-nick').click\_link(user.nick)

# sleep(1)

# find('#nav-myacc').click\_link

# sleep(1)

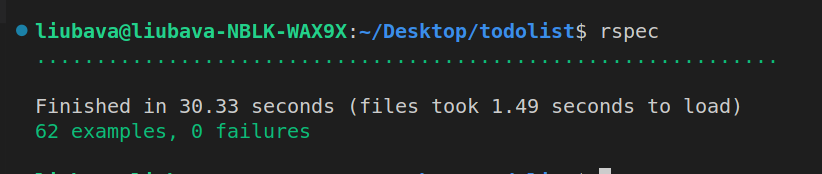
# expect(page).to have\_text(I18n.t(:edit\_account))

# end

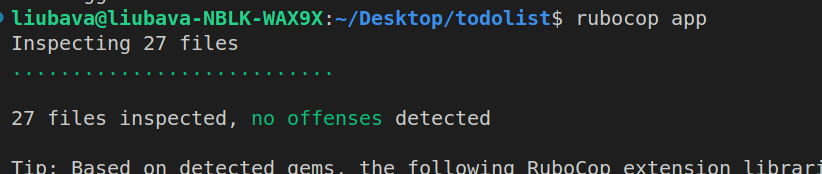
# end

end

**Результаты тестов:**

****

**Проверка на правильность стиля написания кода в соответствии с требованиями Rubocop.**

****

**Вывод.**  Были достигнуты поставленные цели , также применены все знания, полученные за курс.