------------

File: public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width,initial-scale=1.0" />

<link rel="icon" href="<%= BASE\_URL %>favicon.ico" />

<title>vue-chatbot</title>

</head>

<body style="margin: 0;">

<noscript>

<strong>We're sorry but vue-chatbot doesn't work properly without JavaScript

enabled. Please enable it to continue.</strong>

</noscript>

<section style="width: 100%; height: 99vh;">

<iframe src="http://izdo.vvsu.ru/" frameborder="0" width="100%" height="100%"></iframe>

</section>

<div id="app"></div>

<!-- built files will be auto injected -->

</body>

</html>

------------

File: src/App.vue

<template>

<div id="app">

<div class="vvsu-chat-bot">

<Chat />

<ChatButton />

<PhoneButton />

<ModalPhone />

</div>

</div>

</template>

<script>

import Chat from "@/components/Chat";

import ChatButton from "@/components/ChatButton";

import PhoneButton from "@/components/PhoneButton";

import ModalPhone from "@/components/ModalPhone";

export default {

name: "App",

components: {

Chat,

ChatButton,

PhoneButton,

ModalPhone

},

};

</script>

<style lang="css"></style>

------------

File: src/api/questions.api.js

import axios from "axios";

import url from "./url";

// экземпляр axios для обращения к api

const http = axios.create({

baseURL: url.domain

});

export const getAll = () => {

return http.post();

};

// экспорт метода getHistory, который принемает sessionId и отправляет post

// запрос на url.paths.history, с атрибутом id

export const getHistory = sessionId => {

return http.post(url.paths.history, {

id: sessionId

});

};

export const back = (sessionId, step) => {

return http.post(url.paths.back, {

id: sessionId,

step: step

});

};

export const getQuestion = sessionId => {

return http.post(url.paths.getQuestion, {

id: sessionId

});

};

export const registerSession = () => {

return http.post(url.paths.registerSession);

};

export const checkSession = sessionId => {

return http.post(url.paths.checkSession, {

id: sessionId

});

};

export const getAllQuestions = () => {

return http.post(url.paths.keys);

};

export const selectAnswer = (sessionId, questionId) => {

return http.post(url.paths.select, {

id: sessionId,

select: questionId

});

};

------------

File: src/api/url.js

// экспорт методов api

export default {

domain: process.env.VUE\_APP\_URL,

paths: {

back: "chat/back",

select: "chat/select",

history: "chat/history",

getQuestion: "chat/get",

checkSession: "session/check",

registerSession: "session/register",

keys: "chat/keys",

}

};

------------

File: src/assets/stylus/\_colors.styl

mainBotBgColor = #fff

mainBlue = #4E8CFF

subMainBlue = #72a1f7

hightlightColor = #FED330

searchInputBG = #EBEBEB

botMessageColor = #e5e5e5c9

disabledLink = #a5a5a5

@font-face

font-family 'RobotoRegular'

src url('~@/assets/fonts/Roboto-Regular.ttf') format('truetype')

------------

File: src/assets/stylus/chat-button.styl

.btn\_\_chat

z-index 1000

width 60px

height 60px

color #fff

font-size 25px

border-radius 50px

outline none !important

background-color mainBlue

transition 0.4s

cursor pointer

display flex

justify-content center

align-items center

border none

position absolute

bottom -80px

right 0

visibility visible

&:hover

transform scale(1.05)

svg.close

width 20px

svg.open

width 31px

.done

display none

------------

File: src/assets/stylus/chat.styl

\*

margin 0

padding 0

box-sizing border-box

a

text-decoration none

.vvsu-chat-bot

position fixed

right 20px

bottom 100px

top auto

max-width 340px

width 90%

z-index 99999

visibility hidden

.chat-bot

border-radius 15px

height calc(100vh - 100px)

max-height 590px

transition opacity 0.2s ease-in

box-shadow 0px 0px 20px 0px #666

background-color mainBotBgColor

opacity 0

display flex

flex-direction column

position relative

z-index 999

transform translateY(50px)

transition all 0.2s ease-in

&.isActive

opacity 1

transform translateY(0px)

visibility visible

&\_\_close-btn

display none

width 20px

height 20px

margin-left auto

margin-right 10px

&\_\_header

border-top-left-radius 15px

border-top-right-radius 15px

display -webkit-flex

display -moz-flex

display -ms-flex

display -o-flex

display flex

padding 10px

align-items center

flex-flow row nowrap

background-color mainBlue

&\_\_img svg

width 50px

height 50px

&\_\_title

font-family 'RobotoRegular'

font-weight 900

font-size 20px

margin-left 12px

color white

&\_\_chat

padding 23px

overflow-x hidden

overflow-y hidden

&\_\_wrapper

height 100%

position relative

display flex

flex 1 1 auto

flex-direction column

justify-content space-between

&::-webkit-scrollbar

width 0px

&\_\_dialog

overflow-x hidden

overflow-y scroll

overflow -moz-scrollbars-none

-ms-overflow-style none

flex 1 1 auto

.text-helper

color #878b90

font-size 14px

line-height 24px

margin 0 auto

&::-webkit-scrollbar

width 0px

ul, ol > li

list-style-type none

.loading

display flex

flex-direction column

align-items center

justify-content center

&\_\_text

font-size 18px

margin-bottom 20px

font-family RobotoRegular

.status

height 100%

display flex

flex-direction column

font-family RobotoRegular

font-size 18px

text-align left

&\_\_title

font-size 50px

font-weight 700

margin-bottom 15px

&\_\_text, &\_\_reloadPage

margin-bottom 5px

font-size 18px

&\_\_errorCode

font-size 14px

font-weight bold

margin-top 10px

&\_\_svg

margin-top 20px

width 90px

align-self center

@media only screen and (max-width: 430px)

.chat-bot

&\_\_chat

padding 10px

------------

File: src/assets/stylus/loader.styl

.loader

display flex

justify-content center

align-items center

.lds-ellipsis

display inline-block

position relative

width 80px

height 80px

.lds-ellipsis div

position absolute

top 33px

width 13px

height 13px

border-radius 50%

background mainBlue

animation-timing-function cubic-bezier(0, 1, 1, 0)

.lds-ellipsis div:nth-child(1)

left 8px

animation lds-ellipsis1 0.6s infinite

.lds-ellipsis div:nth-child(2)

left 8px

animation lds-ellipsis2 0.6s infinite

.lds-ellipsis div:nth-child(3)

left 32px

animation lds-ellipsis2 0.6s infinite

.lds-ellipsis div:nth-child(4)

left 56px

animation lds-ellipsis3 0.6s infinite

@keyframes lds-ellipsis1

0%

transform scale(0)

100%

transform scale(1)

@keyframes lds-ellipsis3

0%

transform scale(1)

100%

transform scale(0)

@keyframes lds-ellipsis2

0%

transform translate(0, 0)

100%

transform translate(24px, 0)

------------

File: src/assets/stylus/message.styl

\*

margin 0

padding 0

box-sizing border-box

a

text-decoration none

.chat-bot\_\_body

display flex

flex-direction column

overflow-x hidden

overflow-y auto

ul, ol > li

list-style-type none

margin 0 !important

.block-message

padding 11px 18px

color black

line-height 20px

word-wrap break-word

position relative

width 85%

border-radius 8px

font-size 14px

font-family 'RobotoRegular'

overflow: hidden

&:not(:last-child)

margin-bottom 20px

&\_\_back

width: 75px

height 30px

margin-left auto

// Компенсация padding'а родителя .block-message

margin-right -18px

margin-bottom -11px

// Закругление такие же, как и у .block-message

border-radius 8px 0px 0px 0px

text-align center

line-height 30px // Центрирование текста

background-color subMainBlue

transition all 0.2s ease

fill #fff

&:hover

cursor pointer

color black

fill black

// Фикс для экранов (ибо у .block-message меняется padding)

@media only screen and (max-width: 430px)

&\_\_back

margin-right -15px

margin-bottom -9px

&\_\_answer

counter-increment list

margin-left 10px

&:not(:last-child)

margin-bottom 10px

&:before

color black

&.disabled

&:before

color disabledLink !important

span

pointer-events none

cursor default

color disabledLink !important

span

color black

display inline

transition all 0.2s ease

cursor pointer

p

display inline

&:hover

color mainBlue

& ^[1]:before

color mainBlue

&\_\_text

margin-bottom 5px

&\_\_footer

display flex

margin-top 10px

&\_\_data

font-size 14px

font-family 'RobotoRegular'

font-style italic

font-weight 400

.block-bot-message

align-self flex-start

background-color botMessageColor

.block-you-message

align-self flex-end

background-color mainBlue

color white

@media only screen and (max-width: 430px)

.block-message

padding 9px 15px

------------

File: src/assets/stylus/modal-phone.styl

.body\_\_phone

align-items center

position fixed

bottom 100px

visibility hidden

.modal\_\_phone

background-color mainBotBgColor

width 340px

padding 32px

border-radius 10px

box-shadow 0 4px 16px

font-family 'RobotoRegular'

letter-spacing 1px

&.isActive

visibility visible

position relative

&\_\_title

text-align center

margin 0 0 23px 0

font-weight normal

font-size 13pt

&\_\_button

padding: 10px 20px;

border none

border-radius 5px

font-family 'RobotoRegular'

letter-spacing 1px

font-size 16xp

color #fff

background-color mainBlue

outline none

cursor pointer

position relative

left 7%

margin-right 10%

&\_\_number

position relative

margin-bottom 32px

&\_\_input

width 100%

padding 0 0 10px 0

border none

border-bottom 1px solid #e0e0e0

background-color transparent

outline none

color #9e9e9e

------------

File: src/assets/stylus/phone-button.styl

.btn\_\_phone

z-index 1000

width 200px

height 60px

color #fff

font-size 25px

border-radius 50px

outline none !important

background-color mainBlue

cursor pointer

display flex

justify-content center

align-items center

border none

position absolute

bottom -80px

right 70px

font-size 12pt

visibility hidden

&.isActive

visibility visible

&:hover

transform scale(1.05)

svg.close

width 20px

svg.open

width 31px

.done

display none

------------

File: src/assets/stylus/search.styl

.search-block

margin-bottom 20px !important

width 100%

.search-block\_\_wrapper

width 100%

height 40px

position relative

.search-block\_\_input

border 1px solid #eee

border-bottom-left-radius 8px

border-bottom-right-radius 8px

border-top-right-radius 8px

border-top-left-radius 8px

width 100%

height 50px

padding 0 12px

box-sizing border-box

position relative

font-size 16px

line-height 1.5

flex 1

background-color searchInputBG

.highlight

background-color rgba(hightlightColor, 1)

.search-block\_\_input::-webkit-search-decoration, .search-block\_\_input::-webkit-search-cancel-button, .search-block\_\_input::-webkit-search-results-button, .search-block\_\_input::-webkit-search-results-decoration

display none

.search-block\_\_input

&:focus, &:active

border-color rgba(0, 0, 0, 0.12)

background-color #fff

outline none

box-shadow 0 2px 2px rgba(0, 0, 0, 0.16)

&::-ms-clear, &::-ms-reveal

display none

width 0

height 0

&::-webkit-input-placeholder, &::-moz-placeholder, &::-ms-input-placeholder, &::placeholder

color #AAAAAA

.list

border-bottom-left-radius 8px

border-bottom-right-radius 8px

margin 0

border 1px solid rgba(0, 0, 0, 0.12)

padding 0

box-sizing border-box

max-height 296px

overflow-y auto

background #fff

list-style none

box-shadow 0 2px 2px rgba(0, 0, 0, 0.16)

&::-webkit-scrollbar

width 0px !important

&\_\_search-item

padding 8px 0

list-style none

&:first-child

margin-top 8px

&:last-child

margin-bottom 8px

&:hover

background-color #eee

cursor pointer

p

margin 0 15px

font-size 14px

font-family 'RobotoRegular'

line-height 20px

&.selected

background-color rgba(mainBlue, 0.3)

.search-block\_\_reset

outline none

display block

font-size 23px

position absolute

top 12px

right 13px

margin 0

border 0

background none

cursor pointer

padding 0

z-index 99

-webkit-user-select none

-moz-user-select none

-ms-user-select none

user-select none

.search-block\_\_input:valid ~ .search-block\_\_reset

display block

-webkit-animation-name reset-in

animation-name reset-in

-webkit-animation-duration 0.15s

animation-duration 0.15s

@keyframes reset-in

0%

-webkit-transform translate3d(-20%, 0, 0)

transform translate3d(-20%, 0, 0)

opacity 0

100%

-webkit-transform none

transform none

opacity 1

@media only screen and (max-width: 430px)

.search-block\_\_wrapper

height 30px

.search-block\_\_input

height 40px

------------

File: src/components/Chat.vue

<template>

<section class="chat-bot" :class="{ isActive: isChatOpen }">

<div class="chat-bot\_\_header">

<span class="chat-bot\_\_header\_\_img">

<svg

version="1.1"

xmlns="http://www.w3.org/2000/svg"

xmlns:xlink="http://www.w3.org/1999/xlink"

x="0px"

y="0px"

viewBox="0 0 188.148 188.148"

style="enable-background:new 0 0 188.148 188.148;"

xml:space="preserve">

<g>

<g>

<defs>

<circle id="SVGID\_1\_" cx="94.074" cy="94.074" r="94.074" />

</defs>

<use

xlink:href="#SVGID\_1\_"

style="overflow:visible;fill:#EB6D4A;"

/>

<clipPath id="SVGID\_2\_">

<use xlink:href="#SVGID\_1\_" style="overflow:visible;" />

</clipPath>

<path

style="clip-path:url(#SVGID\_2\_);fill:#ECC19C;"

d="M135.938,52.784V84.05c0,0.453-0.011,0.905-0.021,1.357

c-0.02,0.936-0.072,1.871-0.154,2.796c-0.02,0.319-0.052,0.637-0.093,0.956c-0.134,1.326-0.318,2.632-0.555,3.928

c-0.083,0.432-0.165,0.853-0.258,1.275c-0.01,0.092-0.03,0.175-0.051,0.257c-0.083,0.432-0.186,0.854-0.288,1.275

c-0.041,0.226-0.103,0.452-0.154,0.679c-0.052,0.195-0.103,0.391-0.164,0.586c-0.217,0.843-0.463,1.666-0.721,2.488

c-0.123,0.391-0.246,0.771-0.38,1.162c-0.011,0.041-0.031,0.082-0.041,0.123c-0.124,0.349-0.247,0.699-0.37,1.048

c-0.041,0.103-0.082,0.205-0.124,0.319c-0.123,0.329-0.246,0.658-0.38,0.987c-0.154,0.401-0.318,0.792-0.483,1.183

c-0.144,0.35-0.298,0.699-0.452,1.049c-0.02,0.062-0.052,0.113-0.072,0.174c-0.175,0.37-0.339,0.741-0.514,1.1

c-0.175,0.38-0.36,0.761-0.556,1.142c-0.37,0.75-0.771,1.501-1.172,2.231c-0.185,0.329-0.37,0.658-0.556,0.987

c-1.583,2.725-3.341,5.285-5.253,7.649c-0.258,0.339-0.524,0.668-0.802,0.987c-0.381,0.462-0.772,0.915-1.162,1.357

c-0.35,0.391-0.699,0.781-1.059,1.162c-0.987,1.059-1.995,2.067-3.023,3.013c-0.205,0.206-0.422,0.391-0.638,0.586

c-0.349,0.319-0.698,0.627-1.059,0.925c-0.277,0.247-0.565,0.483-0.854,0.72c-0.74,0.606-1.48,1.192-2.23,1.737

c-0.309,0.237-0.628,0.463-0.946,0.679c-0.514,0.37-1.038,0.72-1.563,1.059c-0.288,0.195-0.586,0.38-0.885,0.555

c-1.049,0.648-2.107,1.244-3.156,1.758c-0.257,0.133-0.514,0.257-0.771,0.38c-0.258,0.123-0.524,0.247-0.802,0.37

c-0.34,0.154-0.679,0.298-1.018,0.432c-0.134,0.062-0.268,0.113-0.401,0.154c-0.4,0.165-0.812,0.319-1.213,0.452

c-0.504,0.185-1.018,0.339-1.521,0.483c-0.35,0.103-0.709,0.195-1.059,0.277c-0.021,0.011-0.041,0.021-0.072,0.021

c-0.35,0.082-0.699,0.165-1.049,0.226c-0.093,0.021-0.184,0.041-0.277,0.051c-0.288,0.051-0.576,0.103-0.864,0.134

c-0.411,0.062-0.822,0.113-1.233,0.134c-0.143,0.02-0.277,0.031-0.422,0.031c-0.328,0.021-0.657,0.031-0.977,0.031h-0.03

c-0.319,0-0.638-0.01-0.956-0.031c-0.124,0-0.247-0.01-0.371-0.031c-0.287-0.01-0.586-0.042-0.894-0.083

c-0.062,0-0.134-0.01-0.195-0.02c-0.381-0.051-0.761-0.113-1.152-0.185c-0.328-0.051-0.657-0.124-0.997-0.206

c-1.758-0.401-3.567-1.007-5.397-1.82c-1.563-0.689-3.146-1.521-4.719-2.488c-0.792-0.483-1.584-0.997-2.365-1.542

c-0.391-0.278-0.781-0.555-1.172-0.843c-1.224-0.895-2.437-1.871-3.63-2.92c-0.328-0.298-0.657-0.596-0.986-0.905

c-0.329-0.298-0.658-0.606-0.977-0.904c-0.803-0.771-1.584-1.573-2.345-2.406c-0.298-0.318-0.596-0.647-0.895-0.987

c-0.822-0.925-1.624-1.881-2.385-2.879c-0.299-0.36-0.576-0.73-0.854-1.1c-1.038-1.377-2.025-2.817-2.951-4.308

c-0.246-0.401-0.493-0.813-0.729-1.213c-0.288-0.483-0.565-0.976-0.843-1.47c-1.028-1.851-1.975-3.773-2.797-5.758

c-0.123-0.288-0.247-0.576-0.35-0.863c-0.165-0.37-0.309-0.751-0.442-1.121c-0.205-0.545-0.4-1.09-0.586-1.635

c-0.185-0.545-0.359-1.1-0.534-1.655c-0.165-0.535-0.319-1.069-0.463-1.614c-0.165-0.576-0.318-1.162-0.452-1.758

c-0.011-0.062-0.031-0.123-0.041-0.185c-0.195-0.822-0.37-1.655-0.524-2.498c-0.206-1.111-0.37-2.242-0.494-3.383

c-0.051-0.36-0.082-0.72-0.113-1.08c-0.041-0.38-0.071-0.751-0.092-1.131c-0.052-0.689-0.083-1.388-0.104-2.087

c-0.01-0.453-0.021-0.905-0.021-1.357V52.784c0-23.102,18.723-41.834,41.825-41.855h0.041c3.188,0,6.292,0.359,9.273,1.028

c2.725,0.617,5.347,1.501,7.835,2.622c3.393,1.522,6.549,3.475,9.387,5.809c2.385,1.954,4.555,4.154,6.446,6.58

C132.607,34.083,135.938,43.038,135.938,52.784z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#ECC19C;"

d="M148.613,200.741v0.01H39.538v-0.01c0-6.158,1.018-12.091,2.91-17.612

c0.339-0.997,0.709-1.984,1.11-2.961v-0.01c0.4-0.977,0.822-1.933,1.274-2.879c1.8-3.794,4.041-7.341,6.642-10.59

c0.658-0.812,1.326-1.604,2.016-2.365c0.339-0.391,0.699-0.771,1.059-1.151c0.36-0.37,0.72-0.74,1.09-1.11

c1.841-1.84,3.814-3.526,5.892-5.079c1.038-0.771,2.087-1.491,3.177-2.19c1.09-0.689,2.2-1.347,3.331-1.964

c0.021-0.02,0.041-0.02,0.041-0.02c0.442-0.236,0.874-0.493,1.306-0.761c0.854-0.514,1.696-1.069,2.499-1.665

c0.411-0.288,0.802-0.586,1.192-0.905c0.925-0.74,1.819-1.532,2.673-2.365c0.277-0.277,0.556-0.555,0.833-0.843

c0.575-0.597,1.12-1.224,1.645-1.861c0.258-0.308,0.515-0.627,0.751-0.946c0.247-0.318,0.483-0.647,0.72-0.977

c1.83-2.55,3.3-5.356,4.38-8.349c1.306-3.64,2.015-7.557,2.015-11.649l7.948,0.041l8.173,0.041c0,4.04,0.689,7.916,1.964,11.515

c1.06,3.002,2.529,5.809,4.329,8.359c1.871,2.652,4.102,5.017,6.631,7.042c0.381,0.309,0.761,0.607,1.142,0.884

c0.37,0.278,0.75,0.545,1.141,0.802c0.083,0.062,0.154,0.113,0.237,0.165c0.37,0.247,0.74,0.494,1.131,0.72

c0.421,0.257,0.854,0.514,1.285,0.751c0.071,0.02,0.134,0.051,0.185,0.093c0.905,0.504,1.8,1.028,2.674,1.573

c0.421,0.257,0.843,0.524,1.254,0.823c0.854,0.555,1.687,1.141,2.498,1.758c0,0.011,0.011,0.011,0.011,0.011h0.01

c2.077,1.552,4.041,3.248,5.881,5.079C142.496,172.015,148.613,185.648,148.613,200.741z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#C7D4E2;"

d="M148.614,200.743H39.539c0-6.159,1.019-12.091,2.91-17.612

c0.339-0.998,0.709-1.984,1.11-2.961c-0.011-0.011,0-0.011,0-0.011c0.391-0.977,0.822-1.933,1.274-2.878

c1.8-3.794,4.041-7.341,6.642-10.59c0.658-0.813,1.327-1.604,2.016-2.365c0.339-0.391,0.699-0.771,1.059-1.151

c0.36-0.37,0.72-0.74,1.09-1.11c1.841-1.84,3.814-3.527,5.892-5.079c1.038-0.771,2.087-1.491,3.177-2.19

c1.09-0.699,2.2-1.357,3.331-1.974c0.021-0.02,0.041-0.01,0.041-0.01v-0.01c0.442-0.247,0.874-0.494,1.306-0.751

c1.285-0.781,2.52-1.645,3.671-2.591c0.01,0.01,0.01,0.01,0.021,0.02c5.325,5.449,12.738,8.831,20.964,8.852h0.082

c8.225,0,15.669-3.393,21.015-8.842l0.011-0.01c0.729,0.606,1.49,1.172,2.272,1.696c0.082,0.061,0.154,0.113,0.236,0.164

c0.37,0.247,0.75,0.483,1.131,0.72c0.422,0.257,0.854,0.504,1.285,0.74c0.072,0.02,0.123,0.061,0.185,0.092

c0.905,0.504,1.8,1.028,2.674,1.583c0.421,0.257,0.843,0.524,1.254,0.823c0.854,0.545,1.687,1.141,2.498,1.758h0.021

c2.088,1.563,4.051,3.259,5.881,5.089C142.497,172.017,148.614,185.65,148.614,200.743z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#543E36;"

d="M135.938,84.05c0,0.453-0.011,0.905-0.021,1.357

c-0.02,0.936-0.072,1.871-0.154,2.796c-0.02,0.319-0.052,0.637-0.093,0.956c-0.134,1.326-0.318,2.632-0.555,3.928

c-0.083,0.432-0.165,0.853-0.258,1.275c-0.01,0.092-0.03,0.175-0.051,0.257c-0.083,0.432-0.186,0.854-0.288,1.275

c-0.052,0.247-0.113,0.483-0.186,0.72c-0.03,0.185-0.082,0.37-0.133,0.545c-0.217,0.843-0.463,1.666-0.721,2.488

c-0.123,0.391-0.246,0.771-0.38,1.162c-0.011,0.041-0.031,0.082-0.041,0.123c-0.124,0.349-0.247,0.699-0.37,1.048

c-0.041,0.103-0.082,0.205-0.124,0.319c-0.123,0.329-0.246,0.658-0.38,0.987c-0.154,0.401-0.318,0.792-0.483,1.183

c-0.144,0.35-0.298,0.699-0.452,1.049c-0.02,0.062-0.052,0.113-0.072,0.174c-0.175,0.37-0.339,0.741-0.514,1.1

c-0.175,0.38-0.36,0.761-0.556,1.142c-0.37,0.75-0.771,1.501-1.172,2.231c-0.185,0.329-0.37,0.658-0.556,0.987

c-1.583,2.725-3.341,5.285-5.253,7.649c-0.258,0.339-0.524,0.668-0.802,0.987c-0.381,0.462-0.772,0.915-1.162,1.357

c-0.35,0.391-0.699,0.781-1.059,1.162c-0.987,1.059-1.995,2.067-3.023,3.013c-0.205,0.206-0.422,0.391-0.638,0.586

c-0.349,0.319-0.698,0.627-1.059,0.925c-0.277,0.247-0.565,0.483-0.854,0.72c-0.74,0.606-1.48,1.192-2.23,1.737

c-0.309,0.237-0.628,0.463-0.946,0.679c-0.514,0.37-1.038,0.72-1.563,1.059c-0.288,0.195-0.586,0.38-0.885,0.555

c-1.049,0.648-2.107,1.244-3.156,1.758c-0.257,0.133-0.514,0.257-0.771,0.38c-0.258,0.123-0.524,0.247-0.802,0.37

c-0.34,0.154-0.679,0.298-1.018,0.432c-0.134,0.062-0.268,0.113-0.401,0.154c-0.4,0.165-0.812,0.319-1.213,0.452

c-0.504,0.185-1.018,0.339-1.521,0.483c-0.35,0.103-0.709,0.195-1.059,0.277c-0.021,0.011-0.041,0.021-0.072,0.021

c-0.35,0.082-0.699,0.165-1.049,0.226c-0.093,0.021-0.184,0.041-0.277,0.051c-0.288,0.051-0.576,0.103-0.864,0.134

c-0.411,0.062-0.822,0.113-1.233,0.134c-0.143,0.02-0.277,0.031-0.422,0.031c-0.328,0.021-0.657,0.031-0.977,0.031h-0.03

c-0.319,0-0.638-0.01-0.956-0.031c-0.124,0-0.247-0.01-0.371-0.031c-0.287-0.01-0.586-0.042-0.894-0.083

c-0.062,0-0.134-0.01-0.195-0.02c-0.381-0.051-0.761-0.113-1.152-0.185c-0.328-0.051-0.657-0.124-0.997-0.206

c-1.758-0.401-3.567-1.007-5.397-1.82c-1.563-0.689-3.146-1.521-4.719-2.488c-0.792-0.473-1.573-0.987-2.365-1.542

c-0.391-0.278-0.781-0.555-1.172-0.843c-1.224-0.895-2.437-1.871-3.63-2.92c-0.328-0.298-0.657-0.596-0.986-0.905

c-0.329-0.298-0.658-0.606-0.977-0.904c-0.803-0.771-1.584-1.573-2.345-2.406c-0.298-0.318-0.596-0.647-0.895-0.987

c-0.822-0.925-1.624-1.881-2.385-2.879c-0.299-0.36-0.576-0.73-0.854-1.1c-1.038-1.377-2.025-2.817-2.951-4.308

c-0.246-0.401-0.493-0.813-0.729-1.213c-0.288-0.483-0.565-0.976-0.843-1.47c-1.028-1.851-1.975-3.773-2.797-5.758

c-0.123-0.288-0.247-0.576-0.35-0.863c-0.165-0.37-0.309-0.751-0.442-1.121c-0.205-0.545-0.4-1.09-0.586-1.635

c-0.185-0.545-0.359-1.1-0.534-1.655c-0.165-0.535-0.319-1.069-0.463-1.614c-0.165-0.576-0.318-1.162-0.452-1.758

c-0.011-0.062-0.031-0.123-0.041-0.185c-0.195-0.822-0.37-1.655-0.524-2.498c-0.206-1.111-0.37-2.242-0.494-3.383

c-0.051-0.36-0.082-0.72-0.113-1.08c-0.041-0.38-0.071-0.751-0.092-1.131c-0.052-0.689-0.083-1.388-0.104-2.087

c-0.01-0.453-0.021-0.905-0.021-1.357l6.684,13.14l11.484,5.007l13.581,1.224l10.076,0.915h0.062l9.428,0.854l21.005,1.892

L135.938,84.05z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#543E36;"

d="M127.023,26.968c-2.046,4.637-4.709,8.945-7.855,12.852

c-0.185,0.227-0.37,0.453-0.565,0.679c-6.446,7.762-14.897,13.798-24.562,17.324c-2.54,0.936-5.161,1.696-7.855,2.262

c-1.799,0.38-3.629,0.679-5.49,0.895c-2.139,0.236-4.318,0.36-6.528,0.36c-2.529,0-5.007-0.165-7.443-0.483

c-2.386-0.298-4.72-0.74-7.002-1.336c-1.306-0.329-2.581-0.709-3.846-1.131c-1.244-0.411-2.457-0.864-3.66-1.357v-4.246

c0-23.102,18.723-41.834,41.825-41.855h0.041c3.188,0,6.292,0.359,9.273,1.028c2.725,0.617,5.347,1.501,7.835,2.622

c3.393,1.522,6.549,3.475,9.387,5.809C122.962,22.341,125.132,24.542,127.023,26.968z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#543E36;"

d="M135.938,52.784v8.235c-7.136-5.521-13.047-12.502-17.314-20.48

c-0.01-0.01-0.01-0.031-0.021-0.041c-4.194-7.834-6.796-16.625-7.412-25.92c3.393,1.522,6.549,3.475,9.387,5.809

c2.385,1.954,4.555,4.154,6.446,6.58C132.607,34.083,135.938,43.038,135.938,52.784z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#543E36;"

d="M57.12,56.013c-0.441,0.771-0.843,1.563-1.244,2.375

c-0.153,0.318-0.308,0.658-0.462,0.987c-2.025,4.503-3.044,8.924-3.188,12.914c0,0.113-0.011,0.236-0.011,0.35V56.517

c0-0.576,0.021-1.142,0.021-1.697C53.861,55.242,55.485,55.653,57.12,56.013z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#543E36;"

d="M131.054,56.013c0.622,1.093,1.177,2.213,1.71,3.361

c2.088,4.643,3.11,9.178,3.198,13.268V56.512c0-0.567-0.021-1.134-0.021-1.688C134.318,55.243,132.697,55.648,131.054,56.013z"

/>

<path

style="clip-path:url(#SVGID\_2\_);fill:#1A1A1A;"

d="M130.972,60.167c0-0.175-0.134-0.329-0.309-0.35

c-1.48-0.226-8.143-1.142-15.309-1.142c-8.133,0-18.63,2.55-20.46,2.55H93.27c-0.854,0-3.588-0.555-7.084-1.141

c-4.02-0.679-9.037-1.409-13.376-1.409c-5.243,0-10.22,0.494-13.088,0.843c-1.049,0.124-1.82,0.236-2.221,0.298

c-0.175,0.021-0.309,0.175-0.309,0.35l-0.082,4.369c0,0.154,0.093,0.288,0.236,0.35l0.493,0.175

c0.73,0.278,0.669,13.448,4.596,16.738c2.334,1.943,4.411,3.053,13.16,3.053c5.069,0,8.256-1.367,11.279-5.449

c2.18-2.93,4.112-9.86,4.112-9.86c0.586-2.858,2.714-3.084,3.054-3.105c0.021,0,0.041,0,0.041,0c0.071,0,2.468,0.041,3.095,3.105

c0,0,1.933,6.93,4.112,9.86c3.022,4.082,6.21,5.449,11.278,5.449c8.75,0,10.826-1.11,13.16-3.053

c3.928-3.29,3.866-16.46,4.596-16.738l0.494-0.175c0.143-0.062,0.236-0.195,0.236-0.35L130.972,60.167z M86.875,75.753

c-1.285,3.043-4.606,6.971-9.284,7.207c-14.034,0.709-14.682-3.208-15.587-6.313c-0.915-3.105-1.151-5.83-0.792-8.935

c0.37-3.105,1.152-4.925,2.561-5.84c0.606-0.401,1.265-0.761,2.951-1.018c1.511-0.226,3.845-0.36,7.731-0.36

c2.251,0,4.359,0.175,6.24,0.483c4.997,0.802,8.369,2.499,8.369,4.174C89.064,66.294,88.776,71.249,86.875,75.753z

M126.159,76.647c-0.904,3.105-1.552,7.022-15.586,6.313c-4.678-0.236-7.999-4.164-9.284-7.207

c-1.902-4.503-2.189-9.459-2.189-10.6c0-2.303,6.395-4.658,14.609-4.658c8.215,0,9.53,0.617,10.683,1.378

c1.408,0.915,2.189,2.735,2.56,5.84C127.312,70.818,127.075,73.543,126.159,76.647z"

/>

<g style="clip-path:url(#SVGID\_2\_);">

<path

style="fill:#543E36;"

d="M94.123,90.65l-0.011,10.138l-0.01,3.547v2.786l-0.011,9.253v0.01l-0.01-0.01h-0.011l-0.03-0.021

L53.604,95.771c-0.011-0.062-0.031-0.123-0.041-0.185c-0.195-0.822-0.37-1.655-0.524-2.498c-0.206-1.111-0.37-2.242-0.494-3.383

c-0.051-0.36-0.082-0.72-0.113-1.08c-0.041-0.38-0.071-0.751-0.092-1.131c-0.052-0.689-0.083-1.388-0.104-2.087

c-0.01-0.453-0.021-0.905-0.021-1.357V71.764c0,0.175,0.011,0.35,0.011,0.524c0.082,2.971,0.555,6.241,1.47,9.582

c0.71,2.57,1.594,4.945,2.622,7.063v0.01c2.694,5.604,6.303,9.407,9.623,10.024c0.206,0.031,0.422,0.051,0.628,0.072

c1.285,0.144,2.632,0.226,4.02,0.226c5.243,0,9.901-4.215,12.831-5.922c0.021-0.01,0.041-0.021,0.062-0.031

c0.154-0.093,0.309-0.185,0.442-0.278c0.041-0.031,0.082-0.051,0.123-0.072c2.468-1.409,6.025-2.303,9.994-2.313H94.123z"

/>

<path

style="fill:#543E36;"

d="M135.947,71.764l-0.01,12.286c0,3.085-0.288,6.107-0.822,9.038

c-0.217,1.172-0.463,2.333-0.751,3.485l-9.49,13.705l-30.782,6.097h-0.01l-0.011,0.01v-0.01l-0.03-25.727h0.082

c3.969,0.01,7.536,0.905,10.004,2.313c0.041,0.021,0.071,0.041,0.113,0.072c0.144,0.093,0.298,0.185,0.452,0.278

c0.01,0.01,0.03,0.021,0.051,0.031c2.941,1.707,7.588,5.925,12.832,5.925c1.388,0,2.734-0.082,4.019-0.226

c0.216-0.021,0.422-0.041,0.638-0.072c3.311-0.617,6.93-4.421,9.613-10.024v-0.01c1.027-2.118,1.912-4.493,2.621-7.063

c0.926-3.393,1.409-6.703,1.471-9.716C135.947,72.021,135.947,71.897,135.947,71.764z"

/>

</g>

<ellipse

style="clip-path:url(#SVGID\_2\_);fill:#ECC19C;"

cx="94.074"

cy="103.955"

rx="10.255"

ry="3.17"

/>

<g style="clip-path:url(#SVGID\_2\_);">

<path

style="fill:#435363;"

d="M64.131,155.174v56.558h-4.369v-53.37c0.586-0.473,1.172-0.936,1.769-1.378

C62.384,156.347,63.247,155.75,64.131,155.174z"

/>

<polygon

style="fill:#435363;"

points="124.021,155.185 124.021,200.741 124.021,211.732 128.39,211.732 128.39,200.741

128.39,158.372 "

/>

</g>

<g style="clip-path:url(#SVGID\_2\_);">

<g>

<polygon

style="fill:#141720;"

points="82.113,151.713 94.074,166.049 106.036,151.713 "

/>

<path

style="fill:#141720;"

d="M94.074,225.437h-7.073l5.11-65.8c0.215,0.837,3.712,0.837,3.927,0l5.11,65.8H94.074z"

/>

</g>

<g>

<path

style="fill:#E7ECF2;"

d="M94.07,151.638l-0.03,0.051l-0.011,0.021l-4.966,8.338l-4.637,7.772l-12.554-17.447

c0.401-0.288,0.802-0.596,1.183-0.915c0.01,0.01,0.01,0.01,0.021,0.021c2.518-2.005,4.76-4.359,6.621-6.991l14.343,9.13

L94.07,151.638z"

/>

<path

style="fill:#E7ECF2;"

d="M116.278,150.373l-12.554,17.447l-4.637-7.772l-4.977-8.338l-0.041-0.072l14.436-9.191

c1.871,2.652,4.102,5.017,6.631,7.042C115.518,149.797,115.897,150.096,116.278,150.373z"

/>

</g>

</g>

</g>

</g>

</svg>

</span>

<p class="chat-bot\_\_header\_\_title">ВГУЭС Бот</p>

</div>

<div class="chat-bot\_\_chat">

<div class="loading" v-if="!dialog.length">

<p class="loading\_\_text">Загрузка...</p>

<Loader/>

</div>

<div class="status" v-else-if="status.text">

<p class="status\_\_title">Упс...</p>

<p class="status\_\_text">{{ status.text }}</p>

<p class="status\_\_reloadPage">Перезагрузите страницу</p>

<p class="status\_\_errorCode" v-if="status.errorStatusCode">

Код ошибки: {{ status.errorStatusCode }}

</p>

<div class="status\_\_svg">

<svg viewBox="0 0 47 45" fill="none" xmlns="http://www.w3.org/2000/svg">

<rect x="7" y="11" width="33" height="33" rx="8" stroke="black"/>

<path d="M17 9C17 7.89543 17.8954 7 19 7H27C28.1046 7 29 7.89543 29 9V11H17V9Z" stroke="black"/>

<path d="M42 21C43.1046 21 44 21.8954 44 23L44 31C44 32.1046 43.1046 33 42 33L40 33L40 21L42 21Z" stroke="black"/>

<path d="M5 33C3.89543 33 3 32.1046 3 31L3 23C3 21.8954 3.89543 21 5 21L7 21L7 33L5 33Z" stroke="black"/>

<rect x="1" y="26" width="2" height="2" fill="black" stroke="black"/>

<rect x="44" y="26" width="2" height="2" fill="black" stroke="black"/>

<path d="M23 7C23 2 23.5 2 28.5 2" stroke="black"/>

<circle cx="30" cy="3" r="2" fill="black" stroke="black"/>

<path fill-rule="evenodd" clip-rule="evenodd" d="M30 27C32.2091 27 34 25.2091 34 23C34 20.7909 32.2091 19 30 19C27.7909 19 26 20.7909 26 23C26 25.2091 27.7909 27 30 27ZM30 26C31.6569 26 33 24.6569 33 23C33 21.3431 31.6569 20 30 20C28.3431 20 27 21.3431 27 23C27 24.6569 28.3431 26 30 26Z" fill="black"/>

<path d="M33.5 23C33.5 24.933 31.933 26.5 30 26.5V27.5C32.4853 27.5 34.5 25.4853 34.5 23H33.5ZM30 19.5C31.933 19.5 33.5 21.067 33.5 23H34.5C34.5 20.5147 32.4853 18.5 30 18.5V19.5ZM26.5 23C26.5 21.067 28.067 19.5 30 19.5V18.5C27.5147 18.5 25.5 20.5147 25.5 23H26.5ZM30 26.5C28.067 26.5 26.5 24.933 26.5 23H25.5C25.5 25.4853 27.5147 27.5 30 27.5V26.5ZM32.5 23C32.5 24.3807 31.3807 25.5 30 25.5V26.5C31.933 26.5 33.5 24.933 33.5 23H32.5ZM30 20.5C31.3807 20.5 32.5 21.6193 32.5 23H33.5C33.5 21.067 31.933 19.5 30 19.5V20.5ZM27.5 23C27.5 21.6193 28.6193 20.5 30 20.5V19.5C28.067 19.5 26.5 21.067 26.5 23H27.5ZM30 25.5C28.6193 25.5 27.5 24.3807 27.5 23H26.5C26.5 24.933 28.067 26.5 30 26.5V25.5Z" fill="black"/>

<path fill-rule="evenodd" clip-rule="evenodd" d="M17 20H19V22H18V24H19V26H17V25H15V26H13V24H14V22H13V20H15V21H17V20Z" fill="black"/>

<path d="M19 20H19.5V19.5H19V20ZM17 20V19.5H16.5V20H17ZM19 22V22.5H19.5V22H19ZM18 22V21.5H17.5V22H18ZM18 24H17.5V24.5H18V24ZM19 24H19.5V23.5H19V24ZM19 26V26.5H19.5V26H19ZM17 26H16.5V26.5H17V26ZM17 25H17.5V24.5H17V25ZM15 25V24.5H14.5V25H15ZM15 26V26.5H15.5V26H15ZM13 26H12.5V26.5H13V26ZM13 24V23.5H12.5V24H13ZM14 24V24.5H14.5V24H14ZM14 22H14.5V21.5H14V22ZM13 22H12.5V22.5H13V22ZM13 20V19.5H12.5V20H13ZM15 20H15.5V19.5H15V20ZM15 21H14.5V21.5H15V21ZM17 21V21.5H17.5V21H17ZM19 19.5H17V20.5H19V19.5ZM19.5 22V20H18.5V22H19.5ZM18 22.5H19V21.5H18V22.5ZM18.5 24V22H17.5V24H18.5ZM19 23.5H18V24.5H19V23.5ZM19.5 26V24H18.5V26H19.5ZM17 26.5H19V25.5H17V26.5ZM16.5 25V26H17.5V25H16.5ZM15 25.5H17V24.5H15V25.5ZM15.5 26V25H14.5V26H15.5ZM13 26.5H15V25.5H13V26.5ZM12.5 24V26H13.5V24H12.5ZM14 23.5H13V24.5H14V23.5ZM13.5 22V24H14.5V22H13.5ZM14 21.5H13V22.5H14V21.5ZM13.5 22V20H12.5V22H13.5ZM13 20.5H15V19.5H13V20.5ZM14.5 20V21H15.5V20H14.5ZM17 20.5H15V21.5H17V20.5ZM16.5 20V21H17.5V20H16.5Z" fill="black"/>

<path d="M17 34L17.3123 33.6096C17.1423 33.4735 16.9038 33.4632 16.7226 33.584L17 34ZM19.5 36L19.1877 36.3904C19.3317 36.5057 19.5274 36.5322 19.697 36.4596L19.5 36ZM23 34.5L23.2236 34.0528C23.0921 33.987 22.9382 33.9825 22.803 34.0404L23 34.5ZM26 36L25.7764 36.4472C25.9369 36.5275 26.1281 36.5156 26.2774 36.416L26 36ZM29 34L29.2774 33.584C29.1094 33.472 28.8906 33.472 28.7226 33.584L29 34ZM14.2774 36.416L17.2774 34.416L16.7226 33.584L13.7226 35.584L14.2774 36.416ZM16.6877 34.3904L19.1877 36.3904L19.8123 35.6096L17.3123 33.6096L16.6877 34.3904ZM19.697 36.4596L23.197 34.9596L22.803 34.0404L19.303 35.5404L19.697 36.4596ZM22.7764 34.9472L25.7764 36.4472L26.2236 35.5528L23.2236 34.0528L22.7764 34.9472ZM26.2774 36.416L29.2774 34.416L28.7226 33.584L25.7226 35.584L26.2774 36.416ZM28.7226 34.416L31.7226 36.416L32.2774 35.584L29.2774 33.584L28.7226 34.416Z" fill="black"/>

</svg>

</div>

</div>

<div class="chat-bot\_\_wrapper" v-else>

<Search />

<Dialog :dialog="dialog" />

</div>

</div>

</section>

</template>

<script>

import { mapState, mapActions, mapMutations } from "vuex";

import Search from "@/components/Search.vue";

import Dialog from "@/components/Dialog.vue";

import Loader from "@/components/Loader.vue";

export default {

name: "Chat",

components: {

Search,

Dialog,

Loader

},

computed: {

...mapState(["isChatOpen", "session", "dialog","status"])

},

methods: {

...mapActions([

"FETCH\_QUESTIONS",

"REGISTER\_SESSION",

"GET\_QUESTION",

"CHECK\_SESSION",

"GET\_HISTORY"

]),

...mapMutations(["CLEAR\_COOKIE"]),

},

async created() {

this.FETCH\_QUESTIONS();

// если coockie нет то регестрируем новую сессию и записываем id сессии в куки

if (!this.session) {

// console.log(`кук нет регестрируем сессию`);

await this.REGISTER\_SESSION();

await this.GET\_QUESTION(); // начинаем диалог

} else {

// проверяем на срок годности coockie

const isValid = await this.CHECK\_SESSION();

// если сессия валидна

if (isValid === "valid") {

// подгружаем историю и продолжаем диалог

this.GET\_HISTORY();

} // если сессия не валидна

else {

this.CLEAR\_COOKIE();

}

}

}

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/chat.styl'

</style>

------------

File: src/components/ChatButton.vue

<template>

<button id="btn\_\_chat" type="button" class="btn\_\_chat" @click="openChat()">

<svg

v-if="!isChatOpen && !isModalPhoneOpen"

xmlns="http://www.w3.org/2000/svg"

xmlns:xlink="http://www.w3.org/1999/xlink"

version="1.1"

id="Capa\_1"

x="0px"

y="0px"

viewBox="0 0 511.626 511.626"

style="enable-background:new 0 0 511.626 511.626;"

xml:space="preserve"

class="open">

<path

d="M301.927,327.605c30.926-13.038,55.34-30.785,73.23-53.248c17.888-22.458,26.833-46.915,26.833-73.372 c0-26.458-8.945-50.917-26.84-73.376c-17.888-22.459-42.298-40.208-73.228-53.249c-30.93-13.039-64.571-19.556-100.928-19.556 c-36.354,0-69.995,6.521-100.927,19.556c-30.929,13.04-55.34,30.789-73.229,53.249C8.947,150.072,0,174.527,0,200.986 c0,22.648,6.767,43.975,20.28,63.96c13.512,19.981,32.071,36.829,55.671,50.531c-1.902,4.572-3.853,8.754-5.852,12.566 c-2,3.806-4.377,7.467-7.139,10.991c-2.76,3.525-4.899,6.283-6.423,8.275c-1.523,1.998-3.997,4.812-7.425,8.422 c-3.427,3.617-5.617,5.996-6.567,7.135c0-0.191-0.381,0.24-1.143,1.287c-0.763,1.047-1.191,1.52-1.285,1.431 c-0.096-0.103-0.477,0.373-1.143,1.42c-0.666,1.048-1,1.571-1,1.571l-0.715,1.423c-0.282,0.575-0.476,1.137-0.57,1.712 c-0.096,0.567-0.144,1.19-0.144,1.854s0.094,1.28,0.288,1.854c0.381,2.471,1.475,4.466,3.283,5.996 c1.807,1.52,3.756,2.279,5.852,2.279h0.857c9.515-1.332,17.701-2.854,24.552-4.569c29.312-7.61,55.771-19.797,79.372-36.545 c17.129,3.046,33.879,4.568,50.247,4.568C237.353,347.16,270.998,340.645,301.927,327.605z"

data-original="#000000"

class="active-path"

data-old\_color="#000000"

fill="#FFFFFF"

/>

<path

d="M491.354,338.166c13.518-19.889,20.272-41.247,20.272-64.09c0-23.414-7.146-45.316-21.416-65.68 c-14.277-20.362-33.694-37.305-58.245-50.819c4.374,14.274,6.563,28.739,6.563,43.398c0,25.503-6.368,49.676-19.129,72.519 c-12.752,22.836-31.025,43.01-54.816,60.524c-22.08,15.988-47.205,28.261-75.377,36.829 c-28.164,8.562-57.573,12.848-88.218,12.848c-5.708,0-14.084-0.377-25.122-1.137c38.256,25.119,83.177,37.685,134.756,37.685 c16.371,0,33.119-1.526,50.251-4.571c23.6,16.755,50.06,28.931,79.37,36.549c6.852,1.718,15.037,3.237,24.554,4.568 c2.283,0.191,4.381-0.476,6.283-1.999c1.903-1.522,3.142-3.61,3.71-6.272c-0.089-1.143,0-1.77,0.287-1.861 c0.281-0.09,0.233-0.712-0.144-1.852c-0.376-1.144-0.568-1.715-0.568-1.715l-0.712-1.424c-0.198-0.376-0.52-0.903-0.999-1.567 c-0.476-0.66-0.855-1.143-1.143-1.427c-0.28-0.284-0.705-0.763-1.28-1.424c-0.568-0.66-0.951-1.092-1.143-1.283 c-0.951-1.143-3.139-3.521-6.564-7.139c-3.429-3.613-5.899-6.42-7.422-8.418c-1.523-1.999-3.665-4.757-6.424-8.282 c-2.758-3.518-5.14-7.183-7.139-10.991c-1.998-3.806-3.949-7.995-5.852-12.56C459.289,374.859,477.843,358.062,491.354,338.166z"

data-original="#000000"

class="active-path"

data-old\_color="#000000"

fill="#FFFFFF"

/>

</svg>

<svg

v-else

xmlns="http://www.w3.org/2000/svg"

xmlns:xlink="http://www.w3.org/1999/xlink"

version="1.1"

id="Layer\_1"

x="0px"

y="0px"

viewBox="0 0 492 492"

style="enable-background:new 0 0 492 492;"

xml:space="preserve"

class="close">

<path

d="M300.188,246L484.14,62.04c5.06-5.064,7.852-11.82,7.86-19.024c0-7.208-2.792-13.972-7.86-19.028L468.02,7.872 c-5.068-5.076-11.824-7.856-19.036-7.856c-7.2,0-13.956,2.78-19.024,7.856L246.008,191.82L62.048,7.872 c-5.06-5.076-11.82-7.856-19.028-7.856c-7.2,0-13.96,2.78-19.02,7.856L7.872,23.988c-10.496,10.496-10.496,27.568,0,38.052 L191.828,246L7.872,429.952c-5.064,5.072-7.852,11.828-7.852,19.032c0,7.204,2.788,13.96,7.852,19.028l16.124,16.116 c5.06,5.072,11.824,7.856,19.02,7.856c7.208,0,13.968-2.784,19.028-7.856l183.96-183.952l183.952,183.952 c5.068,5.072,11.824,7.856,19.024,7.856h0.008c7.204,0,13.96-2.784,19.028-7.856l16.12-16.116 c5.06-5.064,7.852-11.824,7.852-19.028c0-7.204-2.792-13.96-7.852-19.028L300.188,246z"

data-original="#000000"

class="active-path"

data-old\_color="#000000"

fill="#FFFFFF"

/>

</svg>

</button>

</template>

<script>

import { mapState, mapMutations } from "vuex";

export default {

name: "ChatButton",

computed: {

...mapState(["isChatOpen", "isShowPhoneButton", "isModalPhoneOpen"])

},

methods: {

...mapMutations(["TOGGLE\_CHAT", "TOGGLE\_PHONEBUTTON", "TOGGLE\_MODAL"]),

openChat() {

this.TOGGLE\_CHAT(!this.isChatOpen);

this.TOGGLE\_PHONEBUTTON(!this.isShowPhoneButton);

if (this.isModalPhoneOpen == true) {

this.TOGGLE\_MODAL(!this.isModalPhoneOpen);

this.TOGGLE\_CHAT(!this.isChatOpen);

}

}

}

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/chat-button.styl'

</style>

------------

File: src/components/Dialog.vue

<template>

<div class="chat-bot\_\_dialog" ref="chatBody">

<ul class="chat-bot\_\_body">

<Message

v-for="(item, index) in dialog"

:key="index + 1"

:message="item"

:selfId="index + 1"/>

<Loader v-if="loadingQuestion"/>

</ul>

</div>

</template>

<script>

import Message from "@/components/Message.vue";

import { mapState } from 'vuex';

import Loader from '@/components/Loader.vue';

export default {

name: "Dialog",

components: {

Message, Loader

},

props: {

dialog: {

type: Array,

required: true,

default: []

}

},

computed: {

...mapState(["loadingQuestion"]),

},

watch: {

dialog() {

// на каждое изменение dialog (в store) список будет прокручиваться вниз

const chatBody = this.$refs.chatBody;

setTimeout(() => {

chatBody.scrollTop = chatBody.scrollHeight;

});

}

},

};

</script>

<style lang="stylus">

.chat-bot

&\_\_dialog

overflow-x hidden

overflow-y scroll

overflow -moz-scrollbars-none

-ms-overflow-style none

flex 1 1 auto

.text-helper

color: #878b90;

font-size: 14px;

line-height: 24px;

margin 0 auto

&::-webkit-scrollbar

width 0px

ul, ol > li

list-style-type none

</style>

------------

File: src/components/Loader.vue

<template>

<div class="loader">

<div class="lds-ellipsis">

<div></div>

<div></div>

<div></div>

<div></div>

</div>

</div>

</template>

<script>

export default {

name: "Loader"

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/loader.styl'

</style>

------------

File: src/components/Message.vue

<template>

<li

class="block-message"

:class="[message.question.user ? 'block-you-message' : 'block-bot-message']"

>

<p class="block-message\_\_text" v-html="message.question.text"></p>

<ul class="block-message\_\_answers" v-if="message.answers">

<li

class="block-message\_\_answer"

:class="disableLinks"

v-for="answer in message.answers"

:key="answer.id"

type="disc"

>

<span

@click="selectAnswer(answer.id, answer.text, message.step)"

v-html="answer.text"

>

</span>

</li>

</ul>

<div class="block-message\_\_footer">

<p class="block-message\_\_footer\_\_data" v-if="!message.question.user">

{{ date }}

</p>

<a

class="block-message\_\_back"

@click="toBackQuestion(selfId)"

v-if="message.question.user"

>

Назад

<svg

xmlns="http://www.w3.org/2000/svg"

xmlns:xlink="http://www.w3.org/1999/xlink"

version="1.1"

id="Capa\_1"

x="0px"

y="0px"

width="13px"

height="10px"

viewBox="0 0 459 459"

style="enable-background:new 0 0 459 459;"

xml:space="preserve"

>

<path

xmlns="http://www.w3.org/2000/svg"

d="M178.5,140.25v-102L0,216.75l178.5,178.5V290.7c127.5,0,216.75,40.8,280.5,130.05C433.5,293.25,357,165.75,178.5,140.25z"

/>

</svg>

</a>

</div>

</li>

</template>

<script>

import { mapActions, mapMutations } from "vuex";

export default {

name: "Message",

// аргументы которые принимает компонент

props: {

message: {

type: Object,

required: true,

default: {}

},

selfId: {

type: Number,

required: true,

default: 0

}

},

data() {

return {};

},

computed: {

disableLinks() {

// если selfId сообщения равен последнему элементу в массиве dialog

// то значит это последнее сообщения (от бота) (сообщения бота всегда последнии)

return this.selfId === this.$store.state.dialog.length ? "" : "disabled";

},

date() {

return this.message.date

? this.message.date

: this.$options.filters.timeFilter(new Date());

}

},

methods: {

...mapMutations(["SET\_DIALOG"]),

...mapActions(["SELECT\_ANSWER", "BACK"]),

toBackQuestion(selfId) {

this.BACK(selfId);

},

selectAnswer(answerId, answerText, messageStep) {

// формируем сообщение юзера

const question = {

question: {

id: answerId,

text: answerText,

user: true

},

step: messageStep

};

// отправляем сообщение юзера в диалог

this.SET\_DIALOG(question);

// вызываем следующий вопрос

this.SELECT\_ANSWER(answerId);

}

}

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/message.styl'

</style>

------------

File: src/components/ModalPhone.vue

<template>

<div class="body\_\_phone">

<form class="modal\_\_phone" :class="{ isActive: isModalPhoneOpen }">

<h1 class="modal\_\_phone\_\_title">Остались вопросы? Оставьте номер телефона и мы перезвоним вам</h1>

<div class="modal\_\_phone\_\_number">

<input v-model="phone" class="modal\_\_phone\_\_input" placeholder="+79001234567">

</div>

<button type='button' class="modal\_\_phone\_\_button" @click="sendPhone">Отправить</button>

<button type='button' class="modal\_\_phone\_\_button" @click="cancel">Отмена</button>

</form>

</div>

</template>

<script>

import { mapState, mapMutations, mapActions} from "vuex";

export default {

name: "ModalPhone",

computed: {

...mapState(["isModalPhoneOpen", "session"])

},

data() {

return {

phone: null

}

},

methods: {

...mapMutations(["TOGGLE\_MODAL", "TOGGLE\_CHAT"]),

sendPhone() {

fetch(`${process.env.VUE\_APP\_URL}chat/phone/?id=${this.session}&phone=${this.phone}`);

this.phone = null;

this.TOGGLE\_MODAL(!this.isModalPhoneOpen);

this.TOGGLE\_CHAT(!this.isChatOpen);

},

cancel() {

this.phone = null;

this.TOGGLE\_MODAL(!this.isModalPhoneOpen);

this.TOGGLE\_CHAT(!this.isChatOpen);

}

},

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/modal-phone.styl'

</style>

------------

File: src/components/PhoneButton.vue

<template>

<button id="btn\_\_phone" type="button" class="btn\_\_phone" :class="{ isActive: isShowPhoneButton }" @click="openModalPhone()">

Остались вопросы?</button>

</template>

<script>

import { mapState, mapMutations } from "vuex";

export default {

name: "PhoneButton",

computed: {

...mapState(["isShowPhoneButton", "isModalPhoneOpen", "isChatOpen"])

},

methods: {

...mapMutations(["TOGGLE\_MODAL", "TOGGLE\_CHAT"]),

openModalPhone() {

this.TOGGLE\_MODAL(!this.isModalPhoneOpen);

this.TOGGLE\_CHAT(!this.isChatOpen);

}

}

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/phone-button.styl'

</style>

------------

File: src/components/Search.vue

<template>

<div class="search-block">

<div class="search-block\_\_wrapper">

<input type="text"

name="search"

placeholder="Ваш вопрос"

autocomplete="off"

class="search-block\_\_input"

v-model="query"

@input="input()"

@focus="showList = true"/>

<button

type="reset"

class="search-block\_\_reset"

@click="closeList"

v-show="query !== ''">

&times;

</button>

</div>

<div>

<ul class="list" ref="questionsList" v-show="showList">

<li v-for="question in allQuestions"

:key="question.id"

role="option"

class="list\_\_search-item"

:class="{ selected: selectedId === question.id }"

@click="itemClick(question.id)">

<p v-html="highlightText(question.text)"></p>

</li>

</ul>

</div>

</div>

</template>

<script>

import { mapActions, mapState, mapMutations, mapGetters } from "vuex";

import {tokenizeText} from '@/utils/tokenizeText.js';

export default {

name: "Search",

data() {

return {

query: '',

selectedItem: null,

selectedId: 0,

showList: false

};

},

computed: {

...mapState(["allQuestions"]),

...mapGetters(["searchedQuestion","getSearchQuestionById"]),

allQuestions(){

return this.searchedQuestion(this.query);

},

},

methods: {

...mapMutations(["SET\_DIALOG"]),

...mapActions(["SELECT\_ANSWER"]),

highlightText(str) {

// если есть текст и запрос

if (str && this.query) {

var highlight = this.query.toLowerCase();

return str

.replace(

new RegExp(highlight, "ig"),

'<span class="highlight">$&</span>'

);

} else {

return str;

}

},

input(){

tokenizeText(this.query);

},

itemClick(questionId) {

this.showList = false

this.selectedId = questionId;

this.selectedItem = this.getSearchQuestionById(this.selectedId);

// формирую сообщение юзера

const question = {

question: {

id: this.selectedItem.id,

text: this.selectedItem.text,

user: true

},

};

// вставляем сообщение юзера в диалог

this.SET\_DIALOG(question);

// вызываю следующий вопрос

this.SELECT\_ANSWER(this.selectedId);

},

closeList() {

this.query = '';

this.showList = false;

},

}

};

</script>

<style lang="stylus">

@import '~@/assets/stylus/\_colors.styl'

@import '~@/assets/stylus/search.styl'

</style>

------------

File: src/filter/time.filter.js

// фильтр даты, получает дату и формат

export default function timeFilter(value) {

const options = {

hour: "2-digit",

minute: "2-digit",

second: "2-digit"

};

return new Intl.DateTimeFormat("ru-RU", options).format(value);

}

------------

File: src/main.js

import Vue from "vue";

import \* as Sentry from "@sentry/browser";

import { Integrations } from "@sentry/tracing";

import App from "./App.vue";

import store from "./store/index";

import timeFilter from "./filter/time.filter";

Sentry.init({

Vue,

dsn: "https://fe4f7257d085491caab0e7d946a67d5c@dsn.frogling.com/5",

autoSessionTracking: true,

logErrors: true,

integrations: [

new Integrations.BrowserTracing(),

],

tracesSampleRate: 1.0,

});

// подключение фильтра dateFilter по имени dateFilter

Vue.filter('timeFilter',timeFilter);

Vue.config.productionTip = false;

new Vue({

store,

render: h => h(App)

}).$mount("#app");

------------

File: src/store/actions.js

import { normalizeHistory } from "../utils/normalize.histoty";

import \* as api from "../api/questions.api";

export default {

statusHandler(

{ commit },

type,

errorStatusCode = null,

text = "Извините что-то пошло не так. Перезагрузите страницу"

) {

commit("CLEAR\_ALL");

commit("SET\_STATUS", { type, text, errorStatusCode });

},

async FETCH\_QUESTIONS({ commit, dispatch }) {

try {

const res = await api.getAllQuestions();

if (res.status === 200) {

// вызываем мутацию SET\_DIALOG и передаем в нее данные

commit("SET\_ALL\_QUESTIONS", res.data);

} else {

dispatch(

"statusHandler",

"FETCH\_QUESTIONS\_STATUS",

res.status,

"Извините не удалось загрузить вопросы для поиска. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "FETCH\_QUESTIONS\_ERROR");

}

},

async REGISTER\_SESSION({ commit }) {

try {

const res = await api.registerSession();

if (res.status === 200) {

commit("SET\_COOKIE", res.data.id);

} else {

dispatch(

"statusHandler",

"REGISTER\_SESSION\_STATUS",

res.status,

"Извините не удалось зарегестрировать сессию. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "REGISTER\_SESSION\_ERROR");

}

},

async GET\_QUESTION({ state, commit }) {

try {

const sessionId = state.session;

const res = await api.getQuestion(sessionId);

if (res.status === 200) {

commit("SET\_DIALOG", res.data);

} else {

dispatch(

"statusHandler",

"GET\_QUESTION\_STATUS",

res.status,

"Извините не удалось загрузить следующий вопрос. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "GET\_QUESTION\_ERROR");

}

},

async CHECK\_SESSION({ state }) {

try {

const sessionId = state.session;

const res = await api.checkSession(sessionId);

if (res.status === 200) {

return res.data.status;

} else {

dispatch(

"statusHandler",

"CHECK\_SESSION\_STATUS",

res.status,

"Извините не удалось проверить сессию. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "CHECK\_SESSION\_ERROR");

}

},

async GET\_HISTORY({ state, commit }) {

try {

const sessionId = state.session;

const res = await api.getHistory(sessionId);

if (res.status === 200) {

const history = await normalizeHistory(res.data);

commit("SET\_HISTORY", history);

} else {

dispatch(

"statusHandler",

"GET\_HISTORY\_STATUS",

res.status,

"Извините не удалось загрузить сессию. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "GET\_HISTORY\_ERROR");

}

},

// работаем с запросами к api

// первый агрумент context у которого есть state, commit, dispatch, getters

// { state, commit } берем из context только то что нужно

// questionId аргумент который передаем в action

async SELECT\_ANSWER({ state, commit }, questionId) {

try {

commit("SET\_LOADING\_QUESTION", true);

// получаем из state сессию

const sessionId = state.session;

// получаем ответ с api

const res = await api.selectAnswer(sessionId, questionId);

if (res.status === 200) {

// commit вызывает мутацию SET\_DIALOG в которую передаем res.data в качестве payload

commit("SET\_DIALOG", res.data);

} else {

dispatch(

"statusHandler",

"SELECT\_ANSWER\_STATUS",

res.status,

"Извините не удалось выбрать вопрос. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "SELECT\_ANSWER\_ERROR");

} finally {

commit("SET\_LOADING\_QUESTION", false);

}

},

async BACK({ state, commit }, selfId) {

try {

// сообщение бота идущее перед сообщением на котором нажали "назад"

// то к которому нам надо вернутся

const before = state.dialog.length;

commit("BACK", selfId);

const backToElement = selfId - 1;

// кол-во шагов которых надо сделать чтоб вернуться к нужному сообщению

const countBack = (before - state.dialog.length + 1) / 2;

const sessionId = state.session;

const res = await api.back(sessionId, countBack);

if (res.status === 200) {

commit("BACK", backToElement);

} else {

dispatch(

"statusHandler",

"BACK\_STATUS",

res.status,

"Извините не удалось вернуться на предыдущий вопрос. Перезагрузите страницу"

);

}

} catch (e) {

dispatch("statusHandler", "BACK\_ERROR");

}

}

};

------------

File: src/store/getters.js

import tokenize from "@/utils/tokenizeText.js";

export default {

getSearchQuestionById: state => id => {

return state.allQuestions.answers.find(q => q.id === id);

},

searchedQuestion: state => searchQuery => {

if (searchQuery === "") return [];

let questions = state.allQuestions.answers.filter(q => {

// Поиск по тексту вопроса и ключевым словам

var text = q.text + " " + q.keys;

if (text.toLowerCase().indexOf(searchQuery.toLowerCase()) !== -1) {

return q;

}

});

// нахожу совпадения и возвращаю первые 10 совпадений

return questions.slice(0, 10);

},

};

------------

File: src/store/index.js

import Vue from "vue";

import Vuex from "vuex";

import state from './state'

import getters from './getters'

import actions from './actions'

import mutations from './mutations'

Vue.use(Vuex);

// хранилище доступное из любого места приложения

export default new Vuex.Store({

// state - переменные хранилища

state,

// getters - возвращают модифицированные даные

getters,

// mutations - используется только для изменения state

mutations,

// actions - используются для всего асинхронного

actions

});

------------

File: src/store/mutations.js

import VueCookie from "vue-cookie";

export default {

// в каждой мутации state должен идти как первый аргумент

TOGGLE\_CHAT(state, payload) {

// state - переменные хранилища

// payload - данные

state.isChatOpen = payload;

},

TOGGLE\_PHONEBUTTON(state, payload) {

state.isShowPhoneButton = payload;

},

TOGGLE\_MODAL(state, payload) {

state.isModalPhoneOpen = payload;

},

SET\_BOT\_STATUS(state, payload) {

state.botStatus = payload;

},

SET\_COOKIE(state, payload) {

state.session = payload;

VueCookie.set("session", payload, { expires: "3h" });

},

CLEAR\_COOKIE(state) {

state.session = null;

VueCookie.delete("session");

},

CLEAR\_ALL(state) {

state.allQuestions = [];

state.dialog = [];

state.isChatOpen = false;

state.session = null;

VueCookie.delete("session");

state.status = {};

},

SET\_DIALOG(state, payload) {

state.dialog.push(payload);

},

SET\_HISTORY(state, payload) {

state.dialog = payload;

},

SET\_ALL\_QUESTIONS(state, payload) {

state.allQuestions = payload;

},

BACK(state, step) {

state.dialog.splice(step);

},

SET\_STATUS(state, status) {

state.status = status;

},

SET\_LOADING\_QUESTION(state, payload) {

state.loadingQuestion = payload;

}

};

------------

File: src/store/state.js

import VueCookie from "vue-cookie";

export default {

// список вопросов для поиска

allQuestions: [],

// для диалога

dialog: [],

// чат открыт/закрыт

isChatOpen: false,

// кнопка "оставьте номер телефона"

isShowPhoneButton: false,

// модальное окно

isModalPhoneOpen: false,

// сессия

session: VueCookie.get("session") || null,

// статус

status: {},

// status: {text: 'Не удалось загрузить вопросы для поиска', errorStatusCode: 403},

loadingQuestion: false,

};

------------

File: src/utils/normalize.histoty.js

export const normalizeHistory = historyFromApi => {

let history = [];

let userMessage = {};

historyFromApi.history.forEach(mes => {

if (mes.selected) {

userMessage.question = mes.selected;

userMessage.question.user = true;

userMessage.step = mes.step;

userMessage.date = mes.date;

delete mes.selected;

history.push(mes);

history.push(userMessage);

} else {

delete mes.selected;

history.push(mes);

}

userMessage = {};

});

return history;

};

------------

File: src/utils/tokenizeText.js

import Az from "az";

export const tokenizeText = text => {

};

------------

File: phpadmin/api\_gutter.php

<?php

include 'config.php';

if (isset($\_POST['a'])) {

$method = $\_POST['a'];

} else if(isset($\_GET['a'])) {

$method = $\_GET['a'];

} else {

$method = null;

}

// Проверка действительности ключа

function keyCheck($key)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/key/check';

// Параметры передаваемые API

$data = array('key' => $key);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

// Читает содержимое файла в строку

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Вывод списока ключей

function keyList()

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/key/list';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data),

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Удаление ключа

function deleteKey($id)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/key/remove';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key'], 'key\_id' => $id);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Добавление ключа

function addKey()

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/key/create';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data),

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Добавление ответа

function addAnswer($question\_id, $text, $keys)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/add/answer';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['text'] = $text;

if ($keys != "") {

$data['keys'] = $keys;

}

$data['question\_id'] = $question\_id;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Добавление вопроса

function addQuestion($answer\_id, $text)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/add/question';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['text'] = $text;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

$q = json\_decode($result);

$qid = $q->{'question'}->{'id'};

if ($result != FALSE) {

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/edit/answer';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['id'] = $answer\_id;

$data['next\_question\_id'] = $qid;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

}

return $result;

}

// Изменение ответа

function editAnswer($answer\_id, $text, $keys)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/edit/answer';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['text'] = $text;

if ($keys != "") {

$data['keys'] = $keys;

} else {

$data['keys'] = "null";

}

$data['id'] = $answer\_id;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Изменение вопроса

function editQuestion($question\_id, $text)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/edit/question';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['text'] = $text;

$data['id'] = $question\_id;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Удаление ответа

function deleteAnswer($answer\_id)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/remove/answer';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['id'] = $answer\_id;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Удаление вопроса

function deleteQuestion($question\_id)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/chat/remove/question';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

$data['id'] = $question\_id;

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Вывод списка оставленных телефонов и их данных

function phoneList()

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/phone/list';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data),

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Удаление телефонов и их данных

function deletePhone($id)

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/phone/remove';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key'], 'phone\_id' => $id);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data)

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

// Удаление всех данных о телефонах

function deleteAllPhones()

{

global $CONFIG;

// Адрес обращения к API

$url = $CONFIG['api\_url'] . 'admin/phone/remove/all';

// Параметры передаваемые API

$data = array('key' => $CONFIG['api\_key']);

// Настройка http-запроса

$options = array(

'http' => array(

'header' => "Content-type: application/x-www-form-urlencoded\r\n",

'method' => 'POST',

'content' => http\_build\_query($data),

)

);

// Создаёт контекст потока

$context = stream\_context\_create($options);

$result = file\_get\_contents($url, false, $context);

return $result;

}

switch ($method) {

case 'addanswer':

addAnswer($\_POST['question\_id'], $\_POST['text'], $\_POST['keys']);

break;

case 'addquestion':

addQuestion($\_POST['answer\_id'], $\_POST['text']);

break;

case 'editanswer':

editAnswer($\_POST['answer\_id'], $\_POST['text'], $\_POST['keys']);

break;

case 'editquestion':

editQuestion($\_POST['question\_id'], $\_POST['text']);

break;

case 'deleteanswer':

deleteAnswer($\_POST['id']);

break;

case 'deletequestion':

deleteQuestion($\_POST['id']);

break;

case 'addkey':

addKey();

break;

case 'deletekey':

deleteKey($\_GET['id']);

break;

case 'deletephone':

deletePhone($\_POST['id']);

break;

case 'deleteallphones':

deleteAllPhones();

break;

case 'logout':

setcookie('api\_key', '', time() - 3600);

header("Location: auth.php");

exit();

break;

case 'sort\_alphabet':

setcookie('sort', 'alphabet', time() + 3600 \* 24 \* 120);

header("Location: index.php");

exit();

break;

case 'sort\_time':

setcookie('sort', 'time', time() + 3600 \* 24 \* 120);

header("Location: index.php");

exit();

break;

default:

break;

}

------------

File: phpadmin/auth.php

<?php

include 'config.php';

include 'api\_gutter.php';

if (isset($\_COOKIE['api\_key'])){

header("Location: index.php");

exit();

}

if (isset($\_POST['key'])) {

$result = keyCheck($\_POST['key']);

$q = json\_decode($result);

$status = $q->{'status'};

if ($status == 'valid') {

setcookie('api\_key', $\_POST['key'], time()+3600\*24\*120);

header("Location: index.php");

exit();

}

}

?>

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.min.css">

<title>Chatbot admin</title>

</head>

<style>

html, body {

height: 100%;

}

body {

display: flex;

align-items: center;

padding-top: 40px;

padding-bottom: 40px;

background-color: #f5f5f5;

}

.form-sign {

width: 100%;

max-width: 530px;

padding: 15px;

margin: auto;

}

</style>

<body>

<div class="container">

<form method="POST" class="form-sign">

<h1 class="h3 mb-3 fw-normal text-center">VVSU-Chatbot CP</h1>

<label class="mb-1">Admin key: </label>

<input name="key" type="text" class="form-control mb-3" placeholder="KEYCODE" required>

<button class="btn btn-lg btn-primary btn-block" type="submit">Войти</button>

</form>

</div>

</body>

------------

File: phpadmin/index.php

<?php

if (!isset($\_COOKIE['api\_key'])) {

header("Location: auth.php");

exit();

}

include 'config.php';

include 'api\_gutter.php';

$tree = file\_get\_contents($CONFIG['api\_url'] . 'admin/chat/tree?key=' . $CONFIG['api\_key']);

$tree = json\_decode($tree, true);

function array\_sort(&$data)

{

if (!$data['answers']) return;

$ar = array();

$sorted\_answers = array();

// Записываем в новый массив id и text

foreach ($data['answers'] as &$answer) {

if (isset($answer['question'])) {

array\_sort($answer['question']);

}

$ar[] = mb\_strtolower($answer['text']);

}

// Сортировка

asort($ar, SORT\_STRING | SORT\_FLAG\_CASE);

foreach ($ar as $key => $value) {

$sorted\_answers[] = $data['answers'][$key];

}

$data['answers'] = $sorted\_answers;

}

function drawTree($data, $level)

{

echo ('<div class="rounded"><div class="h6 mt-3">Q [' . $data['id'] . ']</div><div>' . $data['text'] . '</div></div> <br>');

echo ('<button type="button" class="btn btn-success btn-sm" data-toggle="modal" data-target="#addAnswerModal" data-id="' . $data['id'] . '"><i class="fa fa-plus-square" aria-hidden="true"></i> Добавить вариант</button>');

echo (' <a class="btn btn-warning btn-sm" data-toggle="modal" data-target="#editQuestionModal" data-id="' . $data['id'] . '" data-text="' . htmlspecialchars($data['text']) . '"><i class="fa fa-pencil" aria-hidden="true"></i> Изменить</a>');

if ($data['id'] != 1) {

echo (' <a class="btn btn-danger btn-sm" data-toggle="modal" data-target="#deleteModal" data-id="' . $data['id'] . '" data-a="deletequestion"><i class="fa fa-trash" aria-hidden="true"></i> Удалить</a>');

}

if (!$data['answers']) return;

if ($level == 0) {

echo ('<ul id="treeUL">');

} else {

echo ('<ul class="nested">');

}

foreach ($data['answers'] as $answer) {

if ($level % 2) {

echo ('<li class="mt-3 border border-danger rounded p-2" style="background-color: #e3e3e3;">');

} else {

echo ('<li class="mt-3 border border-primary rounded p-2" style="background-color: #fff;">');

}

if (isset($answer['question']) && $answer['question']['count'] > 0) {

echo ('<span class="caret" data-id="' . $answer['id'] . '"></span>');

}

echo ('<div>A [' . $answer['id'] . ']</div><div>' . $answer['text'] . '</div> <br>');

echo ('<button type="button" class="btn btn-warning btn-sm" data-toggle="modal" data-target="#editAnswerModal" data-id="' . $answer['id'] . '" data-text="' . htmlspecialchars($answer['text']) . '" data-keys="' . htmlspecialchars($answer['keys']) . '"><i class="fa fa-pencil" aria-hidden="true"></i> Изменить</button>');

echo (' <a class="btn btn-danger btn-sm" data-toggle="modal" data-target="#deleteModal" data-id="' . $answer['id'] . '" data-a="deleteanswer"><i class="fa fa-trash" aria-hidden="true"></i> Удалить</a>');

if (isset($answer['question'])) {

drawTree($answer['question'], $level + 1);

} else {

echo (' <button type="button" class="btn btn-primary btn-sm" data-toggle="modal" data-target="#addQuestionModal" data-id="' . $answer['id'] . '"><i class="fa fa-plus-square" aria-hidden="true"></i> Добавить ответ</button>');

}

echo ('</li>');

}

echo ('</ul>');

}

?>

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.min.css">

<link rel="stylesheet" href="css/fork-awesome.min.css">

<title>Chatbot admin</title>

</head>

<style>

ul,

#treeUL {

list-style-type: none;

}

#treeUL {

margin: 0;

padding: 0;

}

.caret {

cursor: pointer;

user-select: none;

}

.caret::before {

content: "\25B6";

color: black;

display: inline-block;

margin-right: 6px;

}

.caret-down::before {

transform: rotate(90deg);

}

.nested {

display: none;

}

.active {

display: block;

}

</style>

<body>

<?php

$page = "main";

include "menu.php";

?>

<div class="container my-3">

<?php if (isset($\_COOKIE['sort']) && $\_COOKIE['sort'] == 'alphabet') {

echo ('<a type="button" href="?a=sort\_time" class="btn pull-right"><i class="fa fa-sort-amount-asc" aria-hidden="true"></i> Сортировать по времени</a>');

array\_sort($tree['tree']);

} else {

echo ('<a type="button" href="?a=sort\_alphabet" class="btn pull-right"><i class="fa fa-sort-alpha-asc" aria-hidden="true"></i> Сортировать по алфавиту</a>');

}

?>

<h1 class="mb-4">Дерево диалога</h1>

<?php

drawTree($tree['tree'], 0); ?>

</div>

<!-- Modal Add Answers-->

<div class="modal fade" id="addAnswerModal" tabindex="-1">

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

<form method="post" action="/">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Добавление варианта ответа</h5>

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

</div>

<div class="modal-body">

<div>Текст ответа: </div>

<input type="hidden" name="a" value="addanswer">

<input class="data\_id" type="hidden" name="question\_id" value="addanswer">

<textarea style="height: 250px;" name="text" class="form-control" aria-label="With textarea" placeholder="Введите текст" id="addanstext"></textarea>

<div class="col-md">

<div class="mt-3">Ключевые слова: </div>

<div class="form-floating">

<input name="keys" class="form-control" id="floatingInputGrid" placeholder="KEYWORDS" value="">

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Сохранить</button>

</div>

</div>

</form>

</div>

</div>

<!-- Modal Edit Answers-->

<div class="modal fade" id="editAnswerModal" tabindex="-1">

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

<form method="post" action="/">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Редактирование варианта ответа</h5>

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

</div>

<div class="modal-body">

<div>Текст ответа: </div>

<input type="hidden" name="a" value="editanswer">

<input class="data\_id" type="hidden" name="answer\_id" value="">

<textarea style="height: 250px;" name="text" class="form-control data\_text" aria-label="With textarea" id="editanstext"></textarea>

<div class="col-md">

<div class="mt-3">Ключевые слова: </div>

<div class="form-floating">

<input name="keys" class="form-control data\_keys" id="floatingInputGrid" placeholder="KEYWORDS">

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Сохранить</button>

</div>

</div>

</form>

</div>

</div>

<!-- Modal Edit Question-->

<div class="modal fade" id="editQuestionModal" tabindex="-1">

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

<form method="post" action="/">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Редактирование вопроса</h5>

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

</div>

<div class="modal-body">

<div>Текст вопроса: </div>

<input type="hidden" name="a" value="editquestion">

<input class="data\_id" type="hidden" name="question\_id" value="">

<textarea style="height: 250px;" name="text" class="form-control data\_text" aria-label="With textarea" id="editquetext"></textarea>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Сохранить</button>

</div>

</div>

</form>

</div>

</div>

<!-- Modal Add Question-->

<div class="modal fade" id="addQuestionModal" tabindex="-1">

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

<form method="post" action="/">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Добавление ответа</h5>

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

</div>

<div class="modal-body">

<label>Текст ответа: </label>

<input type="hidden" name="a" value="addquestion">

<input class="data\_id" type="hidden" name="answer\_id" value="">

<textarea style="height: 250px;" name="text" class="form-control data\_text" aria-label="With textarea" placeholder="Введите текст" id="addquetext"></textarea>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Сохранить</button>

</div>

</div>

</form>

</div>

</div>

<!-- Modal del -->

<div class="modal fade" id="deleteModal" tabindex="-1">

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

<form method="post" action="/">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Удаление</h5>

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

</div>

<div class="modal-body">

<p>Вы уверены что хотите удалить этот вопрос?</p>

<input class="data\_a" type="hidden" name="a" value="">

<input class="data\_id" type="hidden" name="id" value="">

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-danger">Удалить</button>

</div>

</div>

</form>

</div>

</div>

<script src="js/bootstrap.bundle.min.js"></script>

<script src="js/jquery.min.js"></script>

<script>

var toggler = document.getElementsByClassName("caret");

var i;

for (i = 0; i < toggler.length; i++) {

var t = toggler[i];

var saveState = localStorage.getItem("cpchatbot-caret-" + t.getAttribute('data-id'));

if (saveState && saveState == "true") {

t.parentElement.querySelector(".nested").classList.toggle("active");

t.classList.toggle("caret-down");

}

t.addEventListener("click", function() {

if ($(this).hasClass("caret-down")) {

// Find lists

$(this.parentElement).children(".nested").children().each(function(i, elem) {

// Find opened childrens

var caret = $(elem).children(".caret-down").each(function(i, elem) {

// Send event click

elem.click();

});

});

this.parentElement.querySelector(".nested").classList.toggle("active");

this.classList.toggle("caret-down");

localStorage.setItem("cpchatbot-caret-" + this.getAttribute('data-id'), this.classList.contains("caret-down"));

} else {

this.parentElement.querySelector(".nested").classList.toggle("active");

this.classList.toggle("caret-down");

localStorage.setItem("cpchatbot-caret-" + this.getAttribute('data-id'), this.classList.contains("caret-down"));

}

});

}

// For add answer

var addAnswerModal = document.getElementById('addAnswerModal')

addAnswerModal.addEventListener('show.bs.modal', function(event) {

var button = event.relatedTarget

var id = button.getAttribute('data-id')

var modalId = addAnswerModal.querySelector('.data\_id')

modalId.value = id

setTimeout(function() {

document.getElementById("addanstext").focus();

}, 500);

})

// For add question

var addQuestionModal = document.getElementById('addQuestionModal')

addQuestionModal.addEventListener('show.bs.modal', function(event) {

var button = event.relatedTarget

var id = button.getAttribute('data-id')

var modalId = addQuestionModal.querySelector('.data\_id')

modalId.value = id

setTimeout(function() {

document.getElementById("addquetext").focus();

}, 500);

})

// For edit answer

var editAnswerModal = document.getElementById('editAnswerModal')

editAnswerModal.addEventListener('show.bs.modal', function(event) {

var button = event.relatedTarget

var id = button.getAttribute('data-id')

var text = button.getAttribute('data-text')

var keys = button.getAttribute('data-keys')

var modalId = editAnswerModal.querySelector('.data\_id')

var modalText = editAnswerModal.querySelector('.data\_text')

var modalKeys = editAnswerModal.querySelector('.data\_keys')

modalId.value = id

modalText.value = text

modalKeys.value = keys

setTimeout(function() {

document.getElementById("editanstext").focus();

}, 500);

})

// For edit question

var editQuestionModal = document.getElementById('editQuestionModal')

editQuestionModal.addEventListener('show.bs.modal', function(event) {

var button = event.relatedTarget

var id = button.getAttribute('data-id')

var text = button.getAttribute('data-text')

var modalId = editQuestionModal.querySelector('.data\_id')

var modalText = editQuestionModal.querySelector('.data\_text')

modalId.value = id

modalText.value = text

setTimeout(function() {

document.getElementById("editquetext").focus();

}, 500);

})

// For delete answer/question

var deleteModal = document.getElementById('deleteModal')

deleteModal.addEventListener('show.bs.modal', function(event) {

var button = event.relatedTarget

var id = button.getAttribute('data-id')

var method = button.getAttribute('data-a')

var modalId = deleteModal.querySelector('.data\_id')

var modalMethod = deleteModal.querySelector('.data\_a')

modalId.value = id

modalMethod.value = method

})

// Save scroll position

$(document).ready(function() {

if (localStorage.getItem("cpchatbot-quote-scroll") != null) {

$(window).scrollTop(localStorage.getItem("cpchatbot-quote-scroll"));

}

$(window).on("scroll", function() {

localStorage.setItem("cpchatbot-quote-scroll", $(window).scrollTop());

});

});

</script>

</body>

</html>

------------

File: phpadmin/keys.php

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.min.css">

<link rel="stylesheet" href="css/fork-awesome.min.css">

<title>Chatbot admin</title>

</head>

<style>

ul,

#treeUL {

list-style-type: none;

}

#treeUL {

margin: 0;

padding: 0;

}

.caret {

cursor: pointer;

user-select: none;

}

.caret::before {

content: "\25B6";

color: black;

display: inline-block;

margin-right: 6px;

}

.caret-down::before {

transform: rotate(90deg);

}

.nested {

display: none;

}

.active {

display: block;

}

</style>

<body>

<?php

include "api\_gutter.php";

include "config.php";

$page = "keys";

include "menu.php";

if (isset($CONFIG['api\_key'])) {

$keys = keyList();

$keys = json\_decode($keys, true);

}

echo ('<br>');

echo ('<div class="container">');

echo ('<table class="table table-hover table-striped align-middle">

<thead class="table">

<tr>

<th scope="col">id</th>

<th scope="col">key</th>

<th scope="col">action</th>

</tr>

</thead>

<tbody>');

foreach ($keys['keys'] as $value) {

echo ('<tr>

<th scope="row">' . $value['id'] . '</th>

<td>' . $value['key'] . '</td>');

if ($value['key'] != $CONFIG['api\_key']) {

echo ('<td><a class="btn btn-danger btn-sm" href="keys.php?a=deletekey&id=' . $value['id'] . '"><i class="fa fa-trash" aria-hidden="true"></i> Удалить</a></td>

</tr>');

} else {

echo ('<td><button disabled class="btn btn-danger btn-sm" href="keys.php?a=' . $value['id'] . '"><i class="fa fa-trash" aria-hidden="true"></i> Удалить</button></td>

</tr>');

}

}

echo ('</tbody>

<caption><a class="btn btn-success btn-sm" href="keys.php?a=addkey"><i class="fa fa-plus-square" aria-hidden="true"></i> Добавить ключ</a></caption>

</table>');

echo ('</div>');

?>

<script src="js/bootstrap.bundle.min.js"></script>

<script src="js/jquery.min.js"></script>

</body>

</html>

------------

File: phpadmin/menu.php

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

<div class="container container-fluid">

<a class="navbar-brand">VVSU-Chatbot CP</a>

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

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

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link <?php if ($page == "main") echo "active" ?>" href="index.php"><i class="fa fa-cubes" aria-hidden="true"></i> Дерево диалога</a>

</li>

<li class="nav-item">

<a class="nav-link <?php if ($page == "keys") echo "active" ?>" href="keys.php"><i class="fa fa-key-modern" aria-hidden="true"></i> Ключи</a>

</li>

<li class="nav-item">

<a class="nav-link <?php if ($page == "phones") echo "active" ?>" href="phones.php"><i class="fa fa-phone" aria-hidden="true"></i> Телефоны</a>

</li>

</ul>

</div>

</div>

<form method="post" action="/">

<input type="hidden" name="a" value="logout">

<button type="submit" href="?a=logout" class="btn btn btn-secondary">Выход</button>

</form>

</div>

</nav>

------------

File: phpadmin/phones.php

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

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

<link rel="stylesheet" href="css/bootstrap.min.css">

<link rel="stylesheet" href="css/fork-awesome.min.css">

<title>Chatbot admin</title>

</head>

<style>

ul,

#treeUL {

list-style-type: none;

}

#treeUL {

margin: 0;

padding: 0;

}

.caret {

cursor: pointer;

user-select: none;

}

.caret::before {

content: "\25B6";

color: black;

display: inline-block;

margin-right: 6px;

}

.caret-down::before {

transform: rotate(90deg);

}

.nested {

display: none;

}

.active {

display: block;

}

</style>

<body>

<?php

$page = "phones";

include "menu.php";

include "api\_gutter.php";

if (isset($CONFIG['api\_key'])) {

$phones = phoneList();

$phones = json\_decode($phones, true);

}

echo ('<br>');

echo ('<div class="container">');

echo ('<table class="table table-hover table-striped table">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Номер телефона</th>

<th scope="col">История</th>

<th scope="col">Дата</th>

<th scope="col"></th>

</tr>

</thead>

<tbody>');

foreach ($phones['phones'] as $key => $value) {

echo (' <tr>

<th scope ="row">' . $key . '</th>

<td>' . $value['phone'] . '</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-toggle="modal" data-target="#ModalHistory' . $value['id'] . '"><i class="fa fa-history" aria-hidden="true"></i> Посмотреть историю</button>

<div class="modal fade" id="ModalHistory' . $value['id'] . '" role="dialog" aria-modal="true" role="dialog" tabindex="-1">

<div class="modal-dialog modal-dialog-scrollable modal-lg" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">История ' . $value['phone'] . '</h5>

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

</div>

<div class="modal-body">

<textarea style="height: 250px;" name="text" class="form-control" readonly>История:

' . $value['history'] . '

</textarea>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

</td>

<td>' . $value['created\_at'] . '</td>

<th><form method="post" action="phones.php">

<input class="data\_a" type="hidden" name="a" value="deletephone">

<input class="phone\_id" type="hidden" name="id" value="' . $value['id'] . '">

<button type="submit" class="btn btn-danger btn-sm"><i class="fa fa-trash" aria-hidden="true"></i> Удалить</button>

</form>

</th>

</tr>');

}

echo ('</tbody>

<caption>

<form method="post" action="phones.php">

<input class="data\_a" type="hidden" name="a" value="deleteallphones">

<button type="submit" class="btn btn-danger btn-sm"><i class="fa fa-trash" aria-hidden="true"></i> Удалить все данные о телефонах</button>

</form>

</caption>

</table>');

echo ('</div>');

?>

<script src="js/bootstrap.bundle.min.js"></script>

<script src="js/jquery.min.js"></script>

</body>

</html>

------------

File: botapi/app/Admin.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Admin extends Model

{

// Название таблицы с ключами администраторов

protected $table = 'admin\_keys';

// Скрываем данные поля для вывода в JSON

protected $hidden = ['created\_at', 'updated\_at'];

}

------------

File: botapi/app/Answer.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

/\*

\* Ответ на вопрос

\*/

class Answer extends Model

{

// Скрываем данные поля для вывода в JSON

protected $hidden = ['created\_at', 'updated\_at', 'question\_id', 'sort'];

// Получение массива ключевых слов

public function keysArray() {

// Возвращаем пустой массив, если нет ключей

if (empty($this->keys)) return array();

return explode(" ", $this->keys);

}

}

------------

File: botapi/app/Console/Commands/ClearSessions.php

<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

use Illuminate\Support\Facades\DB;

use App\Session;

class ClearSessions extends Command

{

/\*\*

\* Название команды

\*

\* @var string

\*/

protected $signature = 'sessions:clear';

/\*\*

\* Описание команды

\*

\* @var string

\*/

protected $description = 'Clear sessions table and remove all histories';

public function \_\_construct()

{

parent::\_\_construct();

}

/\*\*

\* Выполнение команды

\*

\* @return mixed

\*/

public function handle()

{

if ($this->confirm('This command remove ALL sessions and may disrupt the work of current users. Are you sure to continue?')) {

$count = Session::all()->count();

$this->info('Sessions count: ' . $count);

$this->info('Removing...');

DB::table('sessions')->delete();

$this->info('Done');

} else {

$this->info('Canceled');

}

}

}

------------

File: botapi/app/Console/Commands/InfoSessions.php

<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

use App\Session;

use App\History;

class InfoSessions extends Command

{

/\*\*

\* The name and signature of the console command.

\*

\* @var string

\*/

protected $signature = 'sessions:info';

/\*\*

\* The console command description.

\*

\* @var string

\*/

protected $description = 'Get information about sessions table (and histories)';

/\*\*

\* Create a new command instance.

\*

\* @return void

\*/

public function \_\_construct()

{

parent::\_\_construct();

}

/\*\*

\* Execute the console command.

\*

\* @return mixed

\*/

public function handle()

{

$count\_sessions = Session::all()->count();

$this->info('Sessions count: ' . $count\_sessions);

$count\_histories = History::all()->count();

$this->info('Histories count: ' . $count\_histories);

}

}

------------

File: botapi/app/Console/Kernel.php

<?php

namespace App\Console;

use Illuminate\Console\Scheduling\Schedule;

use Illuminate\Foundation\Console\Kernel as ConsoleKernel;

class Kernel extends ConsoleKernel

{

/\*\*

\* Команды для Artisan представленные сервером

\*

\* @var array

\*/

protected $commands = [ ];

/\*\*

\* Запланированное выполнение команд сервером

\*

\* @param \Illuminate\Console\Scheduling\Schedule $schedule

\* @return void

\*/

protected function schedule(Schedule $schedule)

{

// $schedule->command('inspire')

// ->hourly();

}

/\*\*

\* Внешняя регистрация комманд для сервера.

\*

\* @return void

\*/

protected function commands()

{

$this->load(\_\_DIR\_\_.'/Commands');

require base\_path('routes/console.php');

}

}

------------

File: botapi/app/Exceptions/Handler.php

<?php

namespace App\Exceptions;

use Exception;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

class Handler extends ExceptionHandler

{

/\*\*

\* Список исключений, которые будут игнорироваться

\*

\* @var array

\*/

protected $dontReport = [

];

/\*\*

\* A list of the inputs that are never flashed for validation exceptions.

\*

\* @var array

\*/

protected $dontFlash = [

'password',

'password\_confirmation',

];

/\*\*

\* Report or log an exception.

\*

\* @param \Exception $exception

\* @return void

\*/

public function report(Exception $exception)

{

if (app()->bound('sentry') && $this->shouldReport($exception)) {

app('sentry')->captureException($exception);

}

parent::report($exception);

}

/\*\*

\* Вывод исключения в виде HTTP ответа

\*

\* @param \Illuminate\Http\Request $request

\* @param \Exception $exception

\* @return \Illuminate\Http\Response

\*/

public function render($request, Exception $exception)

{

return parent::render($request, $exception);

// JSON ERROR (for now is off)

if ($exception->getStatusCode()) $message = $exception->getStatusCode();

else $message = $exception->getMessage();

return response()->json(['error' => $message]);

}

}

------------

File: botapi/app/History.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

/\*

\* История вопросов пользователя

\*/

class History extends Model

{

// Скрываем данные поля для вывода в JSON

protected $hidden = ['id', 'created\_at', 'updated\_at'];

}

------------

File: botapi/app/Http/Controllers/AdminController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Log;

use App\Admin;

use App\Question;

use App\Answer;

use App\PhoneNumber;

/\*

\* Контроллер для управления чат ботом (нап. через админ панель)

\*/

class AdminController extends Controller

{

/\*\*

\* @api {get} /admin/key/check Проверка ключа

\* @apiName CheckKey

\* @apiGroup Admin

\* @apiDescription Метод проверки ключа администратора

\*

\* @apiParam {String} key Ключ администратора (будет проверен)

\*

\* @apiSuccess {String} status Статус проверки

\* @apiSuccessExample {json} Пример валидного ключа

\* {

\* "status": "valid"

\* }

\* @apiSuccessExample {json} Пример невалидного ключа

\* {

\* "status": "invalid"

\* }

\*/

public function key\_check(Request $request) {

// Получаем id из запроса

$key = $request->input('key');

if (!$key) {

Log::warning('Method admin/key/check, not enough parameters. Where "key"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем экземпляр ключа

$a = Admin::where('key', $key)->first();

// Проверяем наличие ключа в БД

if (!$a) return response()->json(['status' => 'invalid']);

// Если все хорошо, возвращаем valid

return response()->json(['status' => 'valid']);

}

/\*\*

\* @api {get} /admin/key/info Получение информации о ключе

\* @apiName GetKeyInfo

\* @apiGroup Admin

\* @apiDescription Метод получения информации о ключе

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} key\_id Идентификатор ключа, о котором требуется получить информацию

\*

\* @apiSuccess {Key} key Структура ключа

\*/

public function key\_info(Request $request) {

// Получаем key\_id из запроса

$key\_id = intval($request->input('key\_id'));

if (!$key\_id) {

Log::warning('Method admin/key/info, not enough parameters. Where "key\_id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем ключ

$admin = Admin::where('id', $key\_id)->first();

// Если ключ не найден

if (!$admin) {

Log::warning('Method admin/key/info, key with id ' . $key\_id . ' not found');

return response()->json(['error' => 'key not found'], 400);

}

return response()->json(['key' => $admin], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/key/list Получение списка всех ключей

\* @apiName GetKeyList

\* @apiGroup Admin

\* @apiDescription Метод получения информации о всех ключах

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Keys} keys Массив из структур ключа

\*/

public function key\_list(Request $request) {

// Получаем все ключи

$admins = Admin::all();

// Подсчитываем количество таких ключей

$count = Admin::all()->count();

return response()->json(['keys' => $admins, 'count' => $count], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/phone/list Получение списка номеров телефона

\* @apiName GetPhoneList

\* @apiGroup Admin

\* @apiDescription Метод получения списка оставленных телефонов

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Phone} phone Массив из структура номера телефона

\*/

public function phone\_list(Request $request) {

// Получаем все номера телефонов

$phones = PhoneNumber::all();

// Подсчитываем количество таких ключей

$count = PhoneNumber::all()->count();

return response()->json(['phones' => $phones, 'count' => $count], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/phone/remove Удаление номера телефона

\* @apiName RemovePhone

\* @apiGroup Admin

\* @apiDescription Метод удаления номера телефона

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} phone\_id Идентификатор номера телефона, который нужно удалить

\*

\* @apiSuccess {Status} status Успешность операции (ok / fail)

\*/

public function phone\_remove(Request $request) {

// Получаем phone\_id из запроса

$phone\_id = intval($request->input('phone\_id'));

if (!$phone\_id) {

Log::warning('Method admin/phone/remove, not enough parameters. Where "phone\_id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем номер телефона

$phone = PhoneNumber::where('id', $phone\_id)->first();

// Если номер телефона не найден

if (!$phone) {

Log::warning('Method admin/phone/remove, phone with id ' . $phone\_id . ' not found');

return response()->json(['status' => 'fail', 'error' => 'phone not found'], 400);

}

// Удаляем номер телефона

$phone->delete();

// Если все хорошо, возвращаем ok

return response()->json(['status' => 'ok'], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/phone/remove/all Удаление всех номеров телефона

\* @apiName RemovePhoneAll

\* @apiGroup Admin

\* @apiDescription Метод удаления всех номеров телефона

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Status} status Успешность операции (ok / fail)

\*/

public function phone\_remove\_all(Request $request) {

// Удаляем все номера телефонов

PhoneNumber::truncate();

// Если все хорошо, возвращаем ok

return response()->json(['status' => 'ok'], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/key/create Регистрация нового ключа

\* @apiName CreateKey

\* @apiGroup Admin

\* @apiDescription Метод регистрации нового ключа

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Key} key Структура ключа

\*/

public function key\_create(Request $request) {

// Генерируем рандомный ключ

$characters = '0123456789abcdefghijklmnopqrstuvwxyz';

$key = '';

for ($i = 0; $i < 6; $i++) {

$index = rand(0, strlen($characters) - 1);

$key .= $characters[$index];

}

Log::debug('Generate new admin key ['.$key.']');

// Создаем экземпляр ключа

$admin = new Admin;

$admin->key = $key;

// Записываем ключ в БД

$admin->save();

Log::debug('Save admin key to DB ['.$key.']');

return response()->json(['key' => $admin], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/key/remove Удаление ключа

\* @apiName RemoveKey

\* @apiGroup Admin

\* @apiDescription Метод удаления ключа

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} key\_id Идентификатор ключа, который нужно удалить

\*

\* @apiSuccess {Status} status Успешность операции (ok / fail)

\*/

public function key\_remove(Request $request) {

// Получаем key\_id из запроса

$key\_id = intval($request->input('key\_id'));

if (!$key\_id) {

Log::warning('Method admin/key/remove, not enough parameters. Where "key\_id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем ключ

$admin = Admin::where('id', $key\_id)->first();

// Если ключ не найден

if (!$admin) {

Log::warning('Method admin/key/remove, key with id ' . $key\_id . ' not found');

return response()->json(['status' => 'fail', 'error' => 'key not found'], 400);

}

// Удаляем ключ

$admin->delete();

// Если все хорошо, возвращаем ok

return response()->json(['status' => 'ok'], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/tree Получение базы вопросов-ответов в виде дерева

\* @apiName GetChatTree

\* @apiGroup Admin

\* @apiDescription Метод получения всей базы вопросов-ответов в виде дерева

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Tree} tree Деревовидная структура с вопросами и ответами

\*/

public function chat\_tree(Request $request) {

// Получение первого вопроса

$tree = Question::where('id', 1)->first();

// Запускаем развертку дерева (дорогая операция, кстати)

AdminController::chat\_tree\_expand($tree);

return response()->json(['tree' => $tree], 200, [], JSON\_UNESCAPED\_UNICODE);

}

// Рекурсивная функция развертки дерева

private function chat\_tree\_expand($tree) {

// Получение ответов для вопроса

$answers = Answer::where('question\_id', $tree['id'])->get();

// Перебор всех ответов

foreach($answers as $answer) {

// Получение вопроса, на который ссылается ответ

$question = Question::where('id', $answer['next\_question\_id'])->first();

// Если такой вопрос есть, запускем эту же функцию для него

if($question != null)

AdminController::chat\_tree\_expand($question);

// Приписываем вопрос к ответу

$answer['question'] = $question;

}

// Если массив ответов не пустой - припысываем его к вопросу, иначе возвращаем null

if (count($answers))

$tree['answers'] = $answers;

else

$tree['answers'] = null;

// Количество ответов в массиве

$tree['count'] = count($answers);

}

/\*\*

\* @api {get} /admin/chat/questions Получение базы вопросов

\* @apiName GetQuestions

\* @apiGroup Admin

\* @apiDescription Метод получения всей базы вопросов

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Questions} questions Массив структур вопросов c массивами ответов

\*/

public function chat\_questions(Request $request) {

// Получение всех вопросов

$questions = Question::all();

// Перебор

foreach($questions as $question) {

// Получение вопроса, на который ссылается ответ

$answers = Answer::where('question\_id', $question['id'])->get();

// Перебор всех ответов и конвертация ключевых слов в массив ключевых слов

foreach($answers as &$answer) {

$answer->keys = $answer->keysArray();

}

// Приписываем массив ответов к вопросу

$question['answers'] = $answers;

// Записываем количество найденых ответов

$answers\_count = count($answers);

$question['count'] = $answers\_count;

}

// Подсчитываем количество вопросов

$count = count($questions);

return response()->json(['questions' => $questions, 'count' => $count], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/edit/question Редактирование вопроса

\* @apiName EditQuestion

\* @apiGroup Admin

\* @apiDescription Метод редактирования вопроса

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} id Идентификатор вопроса

\* @apiParam {String} text Содержание вопроса

\*

\* @apiSuccess {Question} question Структура добавленного вопроса

\*/

public function chat\_question\_edit(Request $request) {

// Получаем id из запроса

$id = intval($request->input('id'));

if (!$id) {

Log::warning('Method admin/chat/edit/question, not enough parameters. Where "id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем вопрос

$question = Question::where('id', $id)->first();

// Если вопрос не найден

if (!$question) {

Log::warning('Method admin/chat/edit/question, question with id ' . $id . ' not found');

return response()->json(['error' => 'question not found'], 400);

}

// Если задан параметр text, обновляем его

$text = $request->input('text');

if (!empty($text)) {

$question->text = $text;

}

// Сохраняем вопрос

$question->save();

return response()->json(['question' => $question], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/add/question Добавление вопроса

\* @apiName AddQuestion

\* @apiGroup Admin

\* @apiDescription Метод редактирования вопроса

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {String} text Содержание вопроса

\*

\* @apiSuccess {Question} question Структура созданного вопроса

\*/

public function chat\_question\_add(Request $request) {

// Получаем text из запроса

$text = $request->input('text');

if (!$text) {

Log::warning('Method admin/chat/add/question, not enough parameters. Where "text"?');

return response()->json(['error' => 'not enough parameters']);

}

// Создаем экзепляр вопроса

$question = new Question;

// Если задан параметр text, добавляем его

$text = $request->input('text');

if (!empty($text)) {

$question->text = $text;

}

// Записываем вопрос в БД

$question->save();

Log::debug('Save question to DB ['.$question->id.']');

return response()->json(['question' => $question], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/answers Получение базы ответов

\* @apiName GetAnswers

\* @apiGroup Admin

\* @apiDescription Метод получения всей базы ответов

\*

\* @apiParam {String} key Ключ администратора

\*

\* @apiSuccess {Answers} answers Массив структур ответов

\*/

public function chat\_answers(Request $request) {

// Получение всех ответов

$answers = Answer::all();

// Перебор всех ответов и конвертация ключевых слов в массив ключевых слов

foreach($answers as &$answer) {

$answer->keys = $answer->keysArray();

}

// Подсчитываем количество ответов

$count = count($answers);

return response()->json(['answers' => $answers, 'count' => $count], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/edit/answer Редактирование ответа

\* @apiName EditAnswer

\* @apiGroup Admin

\* @apiDescription Метод редактирования ответа

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} id Идентификатор ответа

\* @apiParam {String} text Содержание ответа

\* @apiParam {String} keys Ключевые слова через пробел для поиска (использовать специальное слово "null" для удаления)

\* @apiParam {Int} question\_id Принадлежность к вопросу

\* @apiParam {Int} next\_question\_id К какому вопросу ведет

\* @apiParam {Int} sort Вес при сортировке в выдаче

\*

\* @apiSuccess {Answer} Answer Структура измененного ответа

\*/

public function chat\_answer\_edit(Request $request) {

// Получаем id из запроса

$id = intval($request->input('id'));

if (!$id) {

Log::warning('Method admin/chat/edit/answer, not enough parameters. Where "id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем ответ

$answer = Answer::where('id', $id)->first();

// Если ответ не найден

if (!$answer) {

Log::warning('Method admin/chat/edit/answer, answer with id ' . $id . ' not found');

return response()->json(['error' => 'answer not found'], 400);

}

// Если задан параметр text, обновляем его

$text = $request->input('text');

if (!empty($text)) {

$answer->text = $text;

}

// Если задан параметр question\_id, обновляем его

$question\_id = intval($request->input('question\_id'));

if (!empty($question\_id)) {

$answer->question\_id = $question\_id;

}

// Если задан параметр next\_question\_id, обновляем его

$next\_question\_id = intval($request->input('next\_question\_id'));

if (!empty($next\_question\_id)) {

$answer->next\_question\_id = $next\_question\_id;

}

// Если задан параметр sort, обновляем его

$sort = intval($request->input('sort'));

if (!empty($sort)) {

$answer->sort = $sort;

}

// Если задан параметр keys, обновляем его

$keys = $request->input('keys');

if (!empty($keys)) {

if ($keys == "null") {

// Удаляем ключи при использовании спец слова

$answer->keys = null;

} else {

// Указываем новые ключи

$answer->keys = $keys;

}

}

// Сохраняем ответ

$answer->save();

return response()->json(['answer' => $answer], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/add/answer Добавление ответа

\* @apiName AddAnswer

\* @apiGroup Admin

\* @apiDescription Метод добавления ответа

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {String} text Содержание ответа

\* @apiParam {String} keys Ключевые слова через пробел для поиска

\* @apiParam {Int} question\_id Принадлежность к вопросу

\* @apiParam {Int} next\_question\_id К какому вопросу ведет

\* @apiParam {Int} sort Вес при сортировке в выдаче

\*

\* @apiSuccess {Answer} Answer Структура добавленного ответа

\*/

public function chat\_answer\_add(Request $request) {

// Создаем ответ

$answer = new Answer;

// Если задан параметр text, добавляем его

$text = $request->input('text');

if (!empty($text)) {

$answer->text = $text;

}

// Если задан параметр question\_id, добавляем его

$question\_id = intval($request->input('question\_id'));

if (!empty($question\_id)) {

$answer->question\_id = $question\_id;

}

// Если задан параметр next\_question\_id, добавляем его

$next\_question\_id = intval($request->input('next\_question\_id'));

if (!empty($next\_question\_id)) {

$answer->next\_question\_id = $next\_question\_id;

}

// Если задан параметр sort, добавляем его

$sort = intval($request->input('sort'));

if (!empty($sort)) {

$answer->sort = $sort;

}

// Если задан параметр keys, добавляем его

$keys = $request->input('keys');

if (!empty($keys)) {

$answer->keys = $keys;

}

// Сохраняем ответ

try{

$answer->save();

} catch(\Exception $e) {

Log::warning('Method admin/chat/add/answer has problem on DB add, check please: '.$answer.'');

return response()->json(['error' => 'unrecoverable error, check relative keys'], 400);

}

Log::debug('Save answer to DB ['.$answer->id.']');

return response()->json(['answer' => $answer], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/remove/question Удаление вопроса

\* @apiName RemoveQuestion

\* @apiGroup Admin

\* @apiDescription Метод удаления вопроса

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} id Идентификатор вопроса, который нужно удалить

\*

\* @apiSuccess {Status} status Успешность операции (ok / fail)

\*/

public function chat\_question\_remove(Request $request) {

// Получаем id из запроса

$id = intval($request->input('id'));

if (!$id) {

Log::warning('Method admin/chat/remove/question, not enough parameters. Where "id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем вопрос

$question = Question::where('id', $id)->first();

// Если вопрос не найден

if (!$question) {

Log::warning('Method admin/chat/remove/question, question with id ' . $id . ' not found');

return response()->json(['status' => 'fail', 'error' => 'question not found'], 400);

}

// Удаляем вопрос

$question->delete();

return response()->json(['status' => 'ok'], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /admin/chat/remove/answer Удаление ответа

\* @apiName RemoveAnswer

\* @apiGroup Admin

\* @apiDescription Метод удаления ответа

\*

\* @apiParam {String} key Ключ администратора

\* @apiParam {Int} id Идентификатор ответа, который нужно удалить

\*

\* @apiSuccess {Status} status Успешность операции (ok / fail)

\*/

public function chat\_answer\_remove(Request $request) {

// Получаем id из запроса

$id = intval($request->input('id'));

if (!$id) {

Log::warning('Method admin/chat/remove/answer, not enough parameters. Where "id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем ответ

$answer = Answer::where('id', $id)->first();

// Если ответ не найден

if (!$answer) {

Log::warning('Method admin/chat/remove/answer, answer with id ' . $id . ' not found');

return response()->json(['status' => 'fail', 'error' => 'answer not found'], 400);

}

// Удаляем ответ

$answer->delete();

return response()->json(['status' => 'ok'], 200, [], JSON\_UNESCAPED\_UNICODE);

}

}

------------

File: botapi/app/Http/Controllers/ChatController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Log;

use App\Question;

use App\History;

use App\Answer;

/\*

\* Контроллер для работы с частом

\*/

class ChatController extends Controller

{

/\*\*

\* @api {get} /chat/get Получение диалога

\* @apiName GetChat

\* @apiGroup Chat

\* @apiDescription Метод получения текущего диалога

\*

\* @apiParam {String} id Ключ сессии

\* @apiParamExample {json} Пример запроса

\* {

\* "id": "0bcbdb27-a69f-4be3-b8c5-1ba7eb8a7d83"

\* }

\*

\* @apiSuccess {Question} question Структура вопроса

\* @apiSuccess {Int} step Текущая глубина диалога

\* @apiSuccess {Answers} answers Массив ответов, которые может выбрать пользователь

\* @apiSuccessExample {json} Пример ответа

\* {

\* "question": {

\* "id": 1,

\* "text": "Здравствуйте, чем мы можем вам помочь?",

\* "user": false

\* },

\* "step": 1,

\* "answers": [{

\* "id": 19,

\* "text": "Вопросы по поступлению",

\* "keys": null

\* }]

\* }

\* @apiSuccessExample {json} Пример ответа с просроченной сессией

\* {

\* "error": "session not valid"

\* }

\*/

public function get(Request $request) {

// Получаем сессию из ранее уже вызванного посредника (в роутах)

$s = $request->get('session');

// Получаем последний элемент истории

$current\_history = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->first();

// Подсчитываем количество элементов истории у сессии

$count = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->count();

// Получем данные о вопросы

$question = Question::where('id', $current\_history->question\_id)->first();

// Для идентификации происхождения вопроса на клиенте

$question['user'] = false;

// Получаем привязанные ответы на вопрос

$answers = Answer::where('question\_id', $question->id)->orderBy('sort')->get();

Log::info('Get question '.$question->id.' for ['.$s->session\_id.'] from history, current step '.$count.'');

return response()->json(['question' => $question, 'step' => $count, 'answers' => $answers], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /chat/select Выбор ответа

\* @apiName SelectAnswer

\* @apiGroup Chat

\* @apiDescription Метод выбора ответа

\*

\* @apiParam {String} id Ключ сессии

\* @apiParam {Int} select id ответа answer

\* @apiParamExample {json} Пример запроса

\* {

\* "id": "0bcbdb27-a69f-4be3-b8c5-1ba7eb8a7d83",

\* "select": 19

\* }

\*

\* @apiSuccess {Question} question Структура вопроса

\* @apiSuccess {Int} step Текущая глубина диалога

\* @apiSuccess {Answers} answers Массив ответов, которые может выбрать пользователь

\* @apiSuccessExample {json} Пример ответа

\* {

\* "question": {

\* "id": 17,

\* "text": "Что вас интересует?",

\* "user": false

\* },

\* "step": 1,

\* "answers": [

\* {

\* "id": 20,

\* "text": "Какой распорядок работы приёмной комиссии?",

\* "keys":"распорядок работа комиссия банан"

\* },

\* {

\* "id": 21,

\* "text": "Какие сроки приема документов для поступления?",

\* "keys":"срок прием документ сдача поступление"

\* }]

\* }

\* @apiSuccessExample {json} Пример ответа с просроченной сессией

\* {

\* "error": "session not valid"

\* }

\*/

public function select(Request $request) {

// Получаем сессию из ранее уже вызванного посредника (в роутах)

$s = $request->get('session');

// Получаем id ответа из запроса

$select = intval($request->input('select'));

if (!$select) {

Log::warning('Method chat/select, not enough parameters, session ['.$s->session\_id.']. Where "select"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем ответ из БД

$answer = Answer::where('id', $select)->first();

// Проверяем существование ответа

if (!$answer) {

Log::warning('Method chat/select, incorrect selection, session ['.$s->session\_id.']. Can\'t find answer with id '.$select.'');

return response()->json(['error' => 'incorrect selection'], 400);

}

// Проверяем, указан ли у ответа следующий вопрос

if ($answer->next\_question\_id == null) {

Log::warning('Method chat/select, answer has no next question, session ['.$s->session\_id.']. Answer id is '.$answer->id.'');

return response()->json(['error' => 'answer has no next question'], 400);

}

// Записываем выбранный ответ в историю

// Получаем последний элемент истории

$current\_history = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->first();

// Указываем ответ

$current\_history->answer\_id = $answer->id;

// Сохраняем историю

$current\_history->save();

// Создаем историю

$history = new History;

// Привязываем к сессии

$history->session\_id = $s->id;

// Указываем вопрос

$history->question\_id = $answer->next\_question\_id;

// Сохраняем историю

$history->save();

Log::info('Select answer '.$answer->id.' for ['.$s->session\_id.'] with next question '.$answer->next\_question\_id.'');

return ChatController::get($request);

}

/\*\*

\* @api {get} /chat/back Возврат на вопрос выше

\* @apiName BackChat

\* @apiGroup Chat

\* @apiDescription Метод возврата на вопрос выше

\*

\* @apiParam {String} id Ключ сессии

\* @apiParam {Int} step Количество шагов (по умолчанию - 1)

\* @apiParamExample {json} Пример запроса

\* {

\* "id": "0bcbdb27-a69f-4be3-b8c5-1ba7eb8a7d83",

\* "step": 3

\* }

\*

\* @apiSuccess {Question} question Структура вопроса

\* @apiSuccess {Int} step Текущая глубина диалога

\* @apiSuccess {Answers} answers Массив ответов, которые может выбрать пользователь

\* @apiSuccessExample {json} Пример ответа

\* {

\* "question": {

\* "id": 1,

\* "text": "Здравствуйте, чем мы можем вам помочь?",

\* "user": false

\* },

\* "step": 1,

\* "answers": [{

\* "id": 19,

\* "text": "Вопросы по поступлению",

\* "keys": null

\* }]

\* }

\* @apiSuccessExample {json} Пример ответа с просроченной сессией

\* {

\* "error": "session not valid"

\* }

\*/

public function back(Request $request, $step = 0) {

// Если step тут равен 0, значит его еще не пытались получить из request

if($step == 0) $step = intval($request->input('step', 1));

// step - количесво возвратов

$step--;

// Получаем сессию из ранее уже вызванного посредника (в роутах)

$s = $request->get('session');

// Получаем последний элемент истории

$current\_history = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->first();

// Подсчитываем количество элементов истории у сессии

$count = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->count();

// Удаляем элемент, если он не последний

if ($count > 1) $current\_history->delete();

// Получаем новый последний элемент истории

$new\_history = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->first();

// Удаляем выбор ответа

$new\_history->answer\_id = null;

$new\_history->save();

// Если требуется еще сделать возвраты - вызываем этот метод

if ($step > 0 && $count > 1) {

Log::info('Back question for ['.$s->session\_id.'], current question id '.$new\_history->question\_id.', recursion step '.$step.'');

return ChatController::back($request, $step);

}

// Если возвраты не требуются, выдаем ответ

else {

Log::info('Back question for ['.$s->session\_id.'], current question id '.$new\_history->question\_id.'');

return ChatController::get($request);

}

}

/\*\*

\* @api {get} /chat/history История диалога сессии

\* @apiName HistoryChat

\* @apiGroup Chat

\* @apiDescription Метод получения истории диалога

\*

\* @apiParam {String} id Ключ сессии

\* @apiParamExample {json} Пример запроса

\* {

\* "id": "0bcbdb27-a69f-4be3-b8c5-1ba7eb8a7d83"

\* }

\*

\* @apiSuccess {History} question Структура истории (массив вопросов и ответов)

\* @apiSuccess {Int} step Текущая глубина диалога

\* @apiSuccessExample {json} Пример ответа

\* {

\* "history": [

\* {

\* "question": {

\* "id": 1,

\* "text": "Здравствуйте, чем мы можем вам помочь?",

\* "user": false

\* },

\* "selected": null,

\* "answers": [

\* {

\* "id": 19,

\* "text": "Вопросы по поступлению",

\* "keys": null

\* }],

\* "date": "06.04.2020 20:50:50",

\* "step":1

\* }

\* ],

\* "step":1

\* }

\* @apiSuccessExample {json} Пример ответа с просроченной сессией

\* {

\* "error": "session not valid"

\* }

\*/

public function history(Request $request) {

// Получаем сессию из ранее уже вызванного посредника (в роутах)

$s = $request->get('session');

// Получаем историю

$history = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->get();

// Подсчитываем количество элементов истории у сессии

$count = History::where('session\_id', $s->id)->orderBy('created\_at', 'DESC')->count();

$data = [];

$local\_step = 1;

foreach($history as &$element) {

$info = [];

// Получем данные о вопросе

$question = Question::where('id', $element->question\_id)->first();

$info['question'] = $question;

// Для идентификации происхождения вопроса на клиенте

$info['question']['user'] = false;

// Получаем данные о выбранном ответе

$answer = Answer::where('id', $element->answer\_id)->first();

$info['selected'] = $answer;

// Получаем данные об всех ответах для вопроса

$answers = Answer::where('question\_id', $question->id)->orderBy('sort')->get();

$info['answers'] = $answers;

// Вывод даты вопроса

$info['date'] = $element->created\_at->format('H:i:s');

// Уровень вопроса

$info['step'] = ($count - $local\_step) + 1;

$local\_step++;

// Записываем вопрос и ответ в массив как отдельный элемент

$data[] = $info;

}

Log::info('Get history for ['.$s->session\_id.'], current step '.$count.'');

return response()->json(['history' => array\_reverse($data), 'step' => $count,], 200, [], JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* @api {get} /chat/keys Получение ключевых слов

\* @apiName KeysChat

\* @apiGroup Chat

\* @apiDescription Метод получения всех вопросов и ключевых слов для поиска

\*

\* @apiSuccess {Answers} answers Массив ответов, которые может выбрать пользователь

\* @apiSuccessExample {json} Пример ответа

\* {

\* "answers": [

\* {

\* "id": 20,

\* "text": "Какой распорядок работы приёмной комиссии?",

\* "keys": "распорядок работа комиссия банан"

\* },

\* {

\* "id": 21,

\* "text": "Какие сроки приема документов для поступления?",

\* "keys": "срок прием документ сдача поступление"

\* },

\* {

\* "id":22,

\* "text": "Где можно заполнить заявление о поступлении?",

\* "keys": "заявление поступление заполнить"

\* },

\* {

\* "id": 23,

\* "text":"По какому адресу находится приёмная комиссия?",

\* "keys":"адрес коммисия"

\* }

\* ],

\* "count":4

\* }

\*/

public function keys(Request $request) {

// Получаем все вопросы с ключевыми словами

$answers = Answer::whereNotNull('keys')->get();

// Подсчитываем количество таких вопросов

$count = Answer::whereNotNull('keys')->count();

return response()->json(['answers' => $answers, 'count' => $count], 200, [], JSON\_UNESCAPED\_UNICODE);

}

}

------------

File: botapi/app/Http/Controllers/Controller.php

<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

use Illuminate\Foundation\Bus\DispatchesJobs;

use Illuminate\Foundation\Validation\ValidatesRequests;

use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController

{

use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

}

------------

File: botapi/app/Http/Controllers/PhoneNumberController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Log;

use App\PhoneNumber;

use App\History;

use App\Question;

use App\Answer;

class PhoneNumberController extends Controller

{

/\*\*

\* @api {get} /phone/send Отправка телефона

\* @apiName GetChat

\* @apiGroup Chat

\* @apiDescription Метод отправки телефона (и сохранение истории сессии)

\*

\* @apiParam {String} id Ключ сессии

\* @apiParam {String} phone Номер телефона

\* @apiParamExample {json} Пример запроса

\* {

\* "id": "0bcbdb27-a69f-4be3-b8c5-1ba7eb8a7d83",

\* "phone": "+79841609900"

\* }

\*

\* @apiSuccess {String} status Статус запроса

\* @apiSuccessExample {json} Пример успешного запроса

\* {

\* "status": "ok"

\* }

\* @apiSuccessExample {json} Пример неуспешного запроса

\* {

\* "status": "fail"

\* }

\*/

public function send(Request $request) {

// Получаем сессию из ранее уже вызванного посредника (в роутах)

$s = $request->get('session');

$phone = new PhoneNumber;

$phone->phone = $request->input('phone');

$phone->session\_id = $s->id;

if (!$phone->phone) {

Log::warning('Method chat/phone, not enough parameters. Where "id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

$history = History::where('session\_id', $s->id)->get();

$text = "";

foreach($history as &$element) {

$answer = Answer::where('id', $element->answer\_id)->first();

$question = Question::where('id', $element->question\_id)->first();

$text = $text . "[$element->created\_at] Бот: $question->text\n";

if ($element->answer\_id != null) {

$text = $text . "[$element->created\_at] Пользователь: $answer->text\n";

}

}

$phone->history = $text;

$phone->save();

Log::error('ERR Called method not implemented yet.');

Log::info('Phone added ['.$s->id.']');

return response()->json(['status' => 'true'], 400, [], JSON\_UNESCAPED\_UNICODE);

}

}

------------

File: botapi/app/Http/Controllers/SessionController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Str;

use Illuminate\Support\Facades\Log;

use App\Session;

use App\History;

class SessionController extends Controller

{

/\*\*

\* @api {get} /session/check Проверка сессии

\* @apiName CheckSession

\* @apiGroup Session

\* @apiDescription Метод проверки существования и актуальности сессии

\*

\* @apiParam {String} id Ключ сессии

\* @apiParamExample {json} Пример запроса

\* {

\* "id": "0bcbdb27-a69f-4be3-b8c5-1ba7eb8a7d83"

\* }

\*

\* @apiSuccess {String} status Состояние сессии

\* @apiSuccessExample {json} Пример валидной сессии

\* {

\* "status": "valid"

\* }

\* @apiSuccessExample {json} Пример невалидной сессии

\* {

\* "status": "invalid"

\* }

\*/

public function check(Request $request) {

// Получаем id сесии из запроса

$session\_id = $request->input('id');

if (!$session\_id) {

Log::warning('Method session/check, not enough parameters. Where "id"?');

return response()->json(['error' => 'not enough parameters'], 400);

}

// Получаем экземпляр сессии из БД

$s = Session::where('session\_id', $session\_id)->first();

// Проверяем наличие сессии в БД и актуальность

if (!$s || !$s->valid()) return response()->json(['status' => 'invalid']);

// Если все хорошо, возвращаем valid

return response()->json(['status' => 'valid']);

}

/\*\*

\* @api {get} /session/register Создание сессии

\* @apiName RegisterSession

\* @apiGroup Session

\* @apiDescription Метод создания сессии

\*

\* @apiSuccess {String} id Уникальный ключ сессии

\* @apiSuccessExample {json} Пример успешно созданной сессии

\* {

\* "id": "7e7c4509-2286-477c-b6e5-da078d842e48"

\* }

\*/

public function register() {

// Генерируем уникальный UUID

$session\_id = (string) Str::uuid();

Log::debug('Generate session key ['.$session\_id.']');

// Создаем экземпляр сессии

$session = new Session;

$session->session\_id = $session\_id;

// Записываем сессию в БД

$session->save();

Log::debug('Save to DB for ['.$session\_id.']');

// Создаем историю

$history = new History;

$history->session\_id = $session->id;

// Указываем начальный вопрос

$history->question\_id = 1;

// Сохраняем историю

$history->save();

Log::debug('Save history to DB for ['.$session\_id.']');

Log::info('Registering session ['.$session\_id.']');

// Возвращаем ID сессии после регистрации

return response()->json([

'id' => $session->session\_id,

]);

}

}

------------

File: botapi/app/Http/Controllers/TestController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Jurager\Morphy\Morphy;

class TestController extends Controller

{

// Проверочный метод

public function test\_message(Request $request) {

$message = $request->input('msg');

$morphy = new Morphy('ru');

$message = $morphy->getPseudoRoot('бегут');

//return $message;

return response()->json(['recieved' => $message, 'test' => 'true'], 200, [], JSON\_UNESCAPED\_UNICODE);

}

}

------------

File: botapi/app/Http/Kernel.php

<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel

{

/\*\*

\* Глобальный стек HTTP посредников сервера

\*

\* Эти посредники будут запускаться для каждого запроса к серверу

\*

\* @var array

\*/

protected $middleware = [

//\App\Http\Middleware\TrustProxies::class,

//\App\Http\Middleware\CheckForMaintenanceMode::class,

//\Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,

//\App\Http\Middleware\TrimStrings::class,

//\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,

];

/\*\*

\* Группы посредников для роутов

\*

\* @var array

\*/

protected $middlewareGroups = [

'api' => [

'throttle:200,1',

'bindings',

],

];

/\*\*

\* The application's route middleware.

\*

\* These middleware may be assigned to groups or used individually.

\*

\* @var array

\*/

protected $routeMiddleware = [

//'auth' => \App\Http\Middleware\Authenticate::class,

//'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,

//'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,

//'can' => \Illuminate\Auth\Middleware\Authorize::class,

//'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,

//'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,

//'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,

'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

'validsession' => \App\Http\Middleware\CheckSession::class,

'adminaccess' => \App\Http\Middleware\CheckAdmin::class,

//'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,

];

/\*\*

\* The priority-sorted list of middleware.

\*

\* This forces non-global middleware to always be in the given order.

\*

\* @var array

\*/

protected $middlewarePriority = [

//\Illuminate\Session\Middleware\StartSession::class,

//\Illuminate\View\Middleware\ShareErrorsFromSession::class,

//\App\Http\Middleware\Authenticate::class,

\Illuminate\Routing\Middleware\ThrottleRequests::class,

//\Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

//\Illuminate\Auth\Middleware\Authorize::class,

];

}

------------

File: botapi/app/Http/Middleware/CheckAdmin.php

<?php

namespace App\Http\Middleware;

use Closure;

use App\Admin;

use Illuminate\Support\Facades\Log;

class CheckAdmin

{

/\*\*

\* Проверяем входящий запрос на наличие валидного ключа админа

\*/

public function handle($request, Closure $next)

{

// Получаем key из запроса

$key = $request->input('key');

if(!$key) {

Log::warning('Middleware check admin, not enough parameters. Where "key"?');

return response()->json(['error' => 'key required']);

}

// Получаем информацию о ключе из БД

$a = Admin::where('key', $key)->first();

// Проверяем наличие ключа в БД

if (!$a) {

Log::debug('Middleware check admin abort, ['.$key.'] is invalid.');

return response()->json(['error' => 'key not valid']);

}

// Если все хорошо, пропускаем и записываем ключ в запрос

$request->attributes->add(['admin' => $a]);

return $next($request);

}

}

------------

File: botapi/app/Http/Middleware/CheckSession.php

<?php

namespace App\Http\Middleware;

use Closure;

use App\Session;

use Illuminate\Support\Facades\Log;

class CheckSession

{

/\*\*

\* Проверяем входящий запрос на наличие валидной сессии

\*/

public function handle($request, Closure $next)

{

// Получаем id сесии из запроса

$session\_id = $request->input('id');

if (!$session\_id) {

Log::warning('Middleware check session, not enough parameters. Where "id"?');

return response()->json(['error' => 'session required']);

}

// Получаем экземпляр сессии из БД

$s = Session::where('session\_id', $session\_id)->first();

// Проверяем наличие сессии в БД и актуальность

if (!$s || !$s->valid()) {

Log::debug('Middleware check session abort, ['.$session\_id.'] is invalid.');

return response()->json(['error' => 'session not valid']);

}

// Если все хорошо, пропускаем и записываем экземпляр сессии в запрос

$request->attributes->add(['session' => $s]);

return $next($request);

}

}

------------

File: botapi/app/PhoneNumber.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class PhoneNumber extends Model

{

//

}

------------

File: botapi/app/Providers/RouteServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends ServiceProvider

{

/\*\*

\* This namespace is applied to your controller routes.

\*

\* In addition, it is set as the URL generator's root namespace.

\*

\* @var string

\*/

protected $namespace = 'App\Http\Controllers';

/\*\*

\* Define your route model bindings, pattern filters, etc.

\*

\* @return void

\*/

public function boot()

{

//

parent::boot();

}

/\*\*

\* Define the routes for the application.

\*

\* @return void

\*/

public function map()

{

$this->mapApiRoutes();

//

}

/\*\*

\* Define the "api" routes for the application.

\*

\* These routes are typically stateless.

\*

\* @return void

\*/

protected function mapApiRoutes()

{

Route::prefix('api')

->middleware('api')

->namespace($this->namespace)

->group(base\_path('routes/api.php'));

}

}

------------

File: botapi/app/Question.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

/\*

\* Вопрос (или конечный ответ, если нет привязанных ответов)

\*/

class Question extends Model

{

// Скрываем данные поля для вывода в JSON

protected $hidden = ['created\_at', 'updated\_at'];

}

------------

File: botapi/app/Session.php

<?php

namespace App;

use DateTime;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Support\Facades\Log;

/\*

\* Сессия пользователя

\*/

class Session extends Model

{

// Название таблицы с сессиями

protected $table = 'sessions';

// Проверка актуальности сессии

public function valid() {

// Получаем текущее время

$now = new DateTime('now');

// Смещаем время на 3 часа

$now->modify("-3 hour");

// Сравниваем смещенное время с датой создания сессии

$interval = date\_diff($this->created\_at, $now);

if (!$interval->invert) {

Log::debug('Check registered session status: ['.$this->session\_id.'] is invalid.');

return false;

}

Log::debug('Check registered session status: ['.$this->session\_id.'] is valid.');

return true;

}

}

------------

File: botapi/routes/api.php

<?php

use Illuminate\Http\Request;

/\* Методы для работы с сессией \*/

// Проверка сессии

Route::match(['get', 'post'], 'session/check', 'SessionController@check');

// Создание сессии

Route::match(['get', 'post'], 'session/register', 'SessionController@register');

/\* Методы для работы с чатом \*/

// Получение диалога

Route::match(['get', 'post'], 'chat/get', 'ChatController@get')->middleware('validsession');

// Выбор ответа

Route::match(['get', 'post'], 'chat/select', 'ChatController@select')->middleware('validsession');

// Возврат на вопрос выше

Route::match(['get', 'post'], 'chat/back', 'ChatController@back')->middleware('validsession');

// История диалога сессии

Route::match(['get', 'post'], 'chat/history', 'ChatController@history')->middleware('validsession');

// Получение ключевых слов

Route::match(['get', 'post'], 'chat/keys', 'ChatController@keys');

// Получение номера телефона

Route::match(['get', 'post'], 'chat/phone', 'PhoneNumberController@send')->middleware('validsession');

/\* Методы для управления \*/

// Проверка ключа

Route::match(['get', 'post'], 'admin/key/check', 'AdminController@key\_check');

// Получение информации о ключе

Route::match(['get', 'post'], 'admin/key/info', 'AdminController@key\_info')->middleware('adminaccess');

// Получение списка всех ключей

Route::match(['get', 'post'], 'admin/key/list', 'AdminController@key\_list')->middleware('adminaccess');

// Регистрация нового ключа

Route::match(['get', 'post'], 'admin/key/create', 'AdminController@key\_create')->middleware('adminaccess');

// Удаление ключа

Route::match(['get', 'post'], 'admin/key/remove', 'AdminController@key\_remove')->middleware('adminaccess');

// Получение базы вопросов-ответов в виде дерева

Route::match(['get', 'post'], 'admin/chat/tree', 'AdminController@chat\_tree')->middleware('adminaccess');

// Получение базы вопросов

Route::match(['get', 'post'], 'admin/chat/questions', 'AdminController@chat\_questions')->middleware('adminaccess');

// Редактирование вопроса

Route::match(['get', 'post'], 'admin/chat/edit/question', 'AdminController@chat\_question\_edit')->middleware('adminaccess');

// Добавление вопроса

Route::match(['get', 'post'], 'admin/chat/add/question', 'AdminController@chat\_question\_add')->middleware('adminaccess');

// Получение базы ответов

Route::match(['get', 'post'], 'admin/chat/answers', 'AdminController@chat\_answers')->middleware('adminaccess');

// Редактирование ответа

Route::match(['get', 'post'], 'admin/chat/edit/answer', 'AdminController@chat\_answer\_edit')->middleware('adminaccess');

// Добавление ответа

Route::match(['get', 'post'], 'admin/chat/add/answer', 'AdminController@chat\_answer\_add')->middleware('adminaccess');

// Удаление вопроса

Route::match(['get', 'post'], 'admin/chat/remove/question', 'AdminController@chat\_question\_remove')->middleware('adminaccess');

// Удаление ответа

Route::match(['get', 'post'], 'admin/chat/remove/answer', 'AdminController@chat\_answer\_remove')->middleware('adminaccess');

// Получение списка номеров телефона

Route::match(['get', 'post'], 'admin/phone/list', 'AdminController@phone\_list')->middleware('adminaccess');

// Удаление номера телефона

Route::match(['get', 'post'], 'admin/phone/remove', 'AdminController@phone\_remove')->middleware('adminaccess');

// Удаление всех номеров телефона

Route::match(['get', 'post'], 'admin/phone/remove/all', 'AdminController@phone\_remove\_all')->middleware('adminaccess');

------------

File: botapi/database/migrations/2019\_11\_04\_074556\_create\_sessions\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateSessionsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('sessions', function (Blueprint $table) {

$table->engine = 'InnoDB';

$table->bigIncrements('id');

$table->string('session\_id')->unique();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('sessions');

}

}

------------

File: botapi/database/migrations/2019\_11\_08\_150742\_create\_questions\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateQuestionsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('questions', function (Blueprint $table) {

$table->engine = 'InnoDB';

$table->bigIncrements('id');

$table->longText('text');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('questions');

}

}

------------

File: botapi/database/migrations/2019\_11\_08\_150745\_create\_answers\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAnswersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('answers', function (Blueprint $table) {

$table->engine = 'InnoDB';

$table->bigIncrements('id');

$table->unsignedBigInteger('question\_id')->nullable();

$table->foreign('question\_id')->references('id')->on('questions')->onDelete('cascade');

$table->longText('text');

$table->unsignedBigInteger('next\_question\_id')->nullable();

$table->foreign('next\_question\_id')->references('id')->on('questions')->onDelete('cascade');

$table->unsignedInteger('sort')->default(0);

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('answers');

}

}

------------

File: botapi/database/migrations/2019\_11\_08\_150753\_create\_histories\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateHistoriesTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('histories', function (Blueprint $table) {

$table->engine = 'InnoDB';

$table->bigIncrements('id');

$table->unsignedBigInteger('session\_id');

$table->foreign('session\_id')->references('id')->on('sessions')->onDelete('cascade');

$table->unsignedBigInteger('question\_id');

$table->foreign('question\_id')->references('id')->on('questions')->onDelete('cascade');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('histories');

}

}

------------

File: botapi/database/migrations/2019\_12\_25\_143230\_add\_histories\_answer.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class AddHistoriesAnswer extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::table('histories', function (Blueprint $table) {

// Запись о выбранном вопросе пользователем

$table->unsignedBigInteger('answer\_id')->nullable()->after('question\_id');

$table->foreign('answer\_id')->references('id')->on('answers')->onDelete('cascade');

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::table('histories', function (Blueprint $table) {

$table->dropForeign(['answer\_id']);

$table->dropColumn('answer\_id');

});

}

}

------------

File: botapi/database/migrations/2020\_03\_15\_204825\_add\_keys\_to\_answers.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class AddKeysToAnswers extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::table('answers', function (Blueprint $table) {

// Ключевые слова для вопроса

$table->string('keys')->nullable();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::table('answers', function (Blueprint $table) {

$table->dropColumn('keys');

});

}

}

------------

File: botapi/database/migrations/2020\_04\_11\_180613\_create\_admin\_keys\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAdminKeysTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('admin\_keys', function (Blueprint $table) {

$table->engine = 'InnoDB';

$table->bigIncrements('id');

$table->string('key')->unique();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('admin\_keys');

}

}

------------

File: botapi/database/migrations/2021\_01\_13\_181858\_create\_phone\_numbers\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreatePhoneNumbersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('phone\_numbers', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('phone');

$table->mediumText('history')->nullable();

$table->unsignedBigInteger('session\_id')->nullable();

$table->foreign('session\_id')->references('id')->on('sessions')->onDelete('set null');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('phone\_number');

}

}