**Название проекта:** 'КСИГМА'

**Цель проекта:** Создание веб-сайта для учеников и их репетиторов

**Задачи:**

1. создать форму регистрации
2. создать форму авторизации
3. создать личный кабинет ученика
4. создать личный кабинет учителя (репетитора)
5. добавить возможность для учителя (репетитора) создавать задачи
6. добавить возможность для ученика решать созданные учителем (репетитором) задачи

**Функционал приложения:**

* титульная страница веб-сайта (пока пустует; в будущем планируется использование титульной страницы для размещения новостей и рекламных объявлений),
* регистрация,
* авторизация,
* личный кабинет:

| ЛК ученика | ЛК учителя (репетитора) |
| --- | --- |
| ученик может:   * видеть свое расписание на месяц (реализовано в виде календаря) * просматривать курсы, уроки, решать задачи и видеть правильность своих ответов * (в будущем можно будет загружать и менять фото профиля) | учитель может:   * видеть свое расписание, отсортированное по ученикам, и редактировать его, назначать ученикам уроки * создавать задачи * создавать уроки из набора задач * создавать курсы из набора уроков * (в будущем можно будет загружать и менять фото профиля) * (в будущем можно будет отслеживать успеваемость ученика) |

**Библиотеки, используемые для организации работы сайта:**

aniso8601==9.0.1

certifi==2021.10.8

charset-normalizer==2.0.12

click==8.0.4

colorama==0.4.4

Flask==2.0.3

Flask-JWT-Simple==0.0.3

Flask-Login==0.5.0

flask-ngrok==0.0.25

Flask-RESTful==0.3.9

Flask-SQLAlchemy==2.5.1

Flask-WTF==1.0.0

greenlet==1.1.2

html==1.13

html5==0.0.9

idna==3.3

itsdangerous==2.1.0

Jinja2==3.0.3

lxml==4.8.0

Mako==1.2.0

MarkupSafe==2.1.0

PyJWT==2.3.0

pytz==2021.3

requests==2.27.1

six==1.16.0

SQLAlchemy==1.4.31

SQLAlchemy-serializer==1.4.1

urllib3==1.26.8

waitress==2.1.0

webencodings==0.5.1

Werkzeug==2.0.3

wtf==0.1

WTForms==3.0.1

**классы для календаря:**

**class GregorianCalendar:**

**статические методы класса:**

def setfirstweekday(weekday: int) - на вход принимает число (устанавливает день, с которого начинается неделя)

def previous\_month\_and\_year(year: int, month: int) **-**на вход принимает два числа - год и месяц, возвращает предыдущий год и месяц в виде чисел

def next\_month\_and\_year(year: int, month: int) - на вход принимает два числа - год и месяц, возвращает следующий год и месяц в виде чисел

def current\_date() - возвращает сегодняшнюю дату в формате дд.мм.гггг

def month\_days(year: int, month: int) - возвращает объект класса *календарь* с установленным первым днем недели и годом, месяцем

def month\_days\_with\_weekday(year: int, month: int) - возвращает объект класса календарь с установленным первым днем недели и выходными днями

**class CalendarData**

def \_\_init\_\_(self, data\_folder: str, first\_weekday: int = WEEK\_START\_DAY\_MONDAY) - принимает на вход путь к файлам, в которых хранятся календари, первый день недели, , инициализирует data\_folder, сам календарь, устанавливает календарю первый день недели

def load\_calendar(self, filename: str) - принимает на вход название файла, возвращает данные из файла

статический метод def is\_past(year: int, month: int, current\_year: int, current\_month: int) - на вход на вход принимает три числа - год, месяц и сегодняшний день, возвращает ПРАВДА, если год прошедший, иначе ЛОЖЬ

def tasks\_from\_calendar(self, year: int, month: int, data: Dict) - на вход принимает два числа - год и месяц - и словарь (данные задачи), возвращает словарь задач

def hide\_past\_tasks(self, year: int, month: int, data: Dict) - на вход принимает два числа - год и месяц - и словарь (данные задачи), скрывает прошедшие задачи

def task\_from\_calendar(self, calendar\_id: str, year: int, month: int, day: int, task\_id: int) - на вход принимает название календаря, год, месяц, день,  ID задачи, возвращает задачу

def repetitive\_task\_from\_calendar(self, calendar\_id: str, year: int, month: int, day: int, task\_id: int) - на вход принимает название календаря, год, месяц, день,  ID задачи, возвращает повторяющуюся задачу

статический метод def date\_for\_frontend(year: int, month: int, day: int) - принимает на вход год, месяц, день, возвращает их в красивом формате для фронтенда

def delete\_task(self, calendar\_id: str, year\_str: str, month\_str: str, day\_str: str, task\_id: int) - на вход принимает название календаря, год, месяц, день, id задачи, удаляет задачу

def update\_task\_day(self, calendar\_id: str, year\_str: str, month\_str: str, day\_str: str, task\_id: int, new\_day\_str: str) - обновляет день задачи

def create\_task(

self,  
calendar\_id: str, year: Optional[int],  
month: Optional[int], day: Optional[int],  
title: str, is\_all\_day: bool,  
start\_time: str,  
details: str,  
color: str,  
has\_repetition: bool,  
repetition\_type: Optional[str],  
repetition\_subtype: Optional[str],  
repetition\_value: int,  
end\_time: Optional[str] = None,  
) - принимает на вход название календаря, месяц, день, название задачи, время начала задачи, дополнительные детали, цвет, повторяемость-неповторяемость задачи, тип повторения, подтип повторения, продолжительность повторения и конец времени задачи, создает задачу

**статические методы**

def add\_tasks\_to\_list(tasks: Dict, day\_str: str, month\_str: str, new\_task: Dict) - на вход принимает словарь задач, день, месяц, новую задачу, добавляет в словарь задач новую задачу

def \_clear\_empty\_entires(data: Dict) - принимает на вход словарь, подчищает мусор

ПЕРЕХОДИМ К БАЗЕ ДАННЫХ, ЕЕ СТРУКТУРЕ И ORM-МОДЕЛЯМ

База данных SITE

Таблица users

| ID Integer Primary key Not null |
| --- |
| Name VARCHARunique True |
| Surname VARCHAR(пока не используется) |
| grade Integer |
| email String unique |
| hashed\_password VARCHAR unique |
| created\_date DATETIME unique |
| is\_teacher BOOLEAN |
| courses STRING |

Таблица teachers

| ID Integer Primary key Foreign key(users ID) |
| --- |
| Name String |
| Surname String(пока не используется) |
| Email String |
| Hashed\_password String |
| Created\_date Datetime |
| salary(пока не используется) |
| students String |

Таблица Object

| ID Integer Primary key Not null |
| --- |
| name VARCHAR unique |

Таблица Tasks

| ID Integer Primary key Not null |
| --- |
| name VARCHAR |
| type\_object integer foreign key(object ID) |
| type string |
| question string |
| answers string |
| varations\_of\_answers string |

Таблица lessons

| ID Integer Primary key |
| --- |
| name string |
| tasks string |
| type\_object integer foreign key(object ID) |

Таблица courses

| ID Integer Primary key |
| --- |
| name string |
| object integer foreign key(object ID) |
| lessons string |

Таблица complete\_tasks

| ID Integer foreign key(users ID) |
| --- |
| name string |

Таблица calendars

| student ID Integer foreign key(users ID) |
| --- |
| calendar\_name string |

ORM-МОДЕЛИ аналогичны базам

Работа с БД сделана в файле help\_function.py (Верю в свой нейминг и комментарии)

**ПЕРЕЙДЕМ К САЙТУ**

**Классы:**

class RegisterForm(FlaskForm) - форма для регистрации

class LoginForm(FlaskForm) - форма для входа

class Task(Resource) - api для календаря, создает задачу

class NameOfTask(Resource) - возвращает название задачи, api

class SaveProblem(Resource) - сохраняет задачу, которую загрузил учитель, api

class CheckProblem(Resource) - для проверки задачи, сделанной ученикомб api

class Badlesson(Exception) - класс-исключение, если в уроке не у всех задач одинаковое значение type\_object

class Badcourse(Exception) - класс-исключение, если в курсе не у всех уроков одинаковое значение type\_object

**Функции:**

@login\_manager.user\_loader

def load\_user(user\_id):

db\_sess = db\_session.create\_session()

return db\_sess.query(User).get(user\_id) - на вход принимает ID пользователя, создает сессию (регистрирует в ней пользователя)

@application.errorhandler(werkzeug.exceptions.InternalServerError)  
 def handle\_something\_go\_wrong(e):  
 print(e)  
 return 'we are so sorry, but something go wrong!', 500 - отлавливает ошибку статус 500 и возвращает сообщение об ошибке

@application.errorhandler(werkzeug.exceptions.Forbidden)

def handle\_bad\_request(e):

return 'This page is only for teachers, ha-ha-ha loser!', 400 - отлавливает ошибку статус 400 и возвращает сообщение о нехватке прав для посещения этой страницы

@application.route('/')

def start(): - стартовая страница

@application.route('/register', methods=['GET', 'POST'])

def register(): - страница регистрации (при регистрации создается календарь для ученика и complete\_tasks)

@application.route('/login', methods=['GET', 'POST'])

def login(): - страница входа

@application.route('/logout')

@login\_required

def logout(): - выход из сессии пользователя

@application.route('/cabinet')

@login\_required

def cabinet(): - личный кабинет ученика

@application.route('/teacher\_cabinet')

@login\_required

def teacher\_cabinet(): - личный кабинет учителя (репетитора)

@application.route('/teacher\_calendar/<int:user\_id>/')  
 @login\_required  
 def teacher\_calendar(user\_id): - расписание для учителя(календарь)

@application.route('/teacher/tasks', methods=['POST', 'GET'])

@login\_required

def tasks(): - создание задачи для **календаря**

@application.route('/teacher/check\_tasks', methods=['POST'])  
 @login\_required  
 def check\_tasks(): - проверка созданной задачи учителем

@application.route('/make/<calendar\_id>/<year>/<month>/new\_task/<day\_now>/new\_calendar\_task',

methods=['GET', 'POST'])

def new\_task\_action(calendar\_id: int, year: int, month: int, day\_now: int): - принимает на вход имя календаря, год, месяц, день, создает задачу и возвращает на расписание учителя

@application.route('/<calendar\_id>/<year>/<month>/<day>/<task\_id>/delete\_task',

methods=['POST', 'DELETE', 'GET'])

def delete\_task(calendar\_id, year, month, day, task\_id): - удаляет задачу из календаря

@application.route('/calendar\_student')

def student\_calendar(): - расписание для студента

@application.route('/create\_lesson/<filter\_name>/', methods=["GET", "POST"])

def create\_lesson(filter\_name): - создания урока из задач, созданных учителем

@application.route('/view\_task/<task>/', methods=['POST', 'DELETE', 'GET'])

def view\_task(task): - посмотреть, как примерно будет выглядеть задача

@application.route('/lesson', methods=['POST'])

def lesson(): вспомогательная функция - создает урок

@application.route('/create\_courses/<filter\_name>/', methods=["GET", "POST"])

def create\_cour(filter\_name) - создание курсов из уроков

@application.route('/course', methods=['POST'])

def course(): - вспомогательная функция, создает курс

@application.route('/change\_course/<filter\_name>/')

def change\_course(filter\_name) - изменение доступных курсов(добавление/удаление уроков в/из них)

@application.route('/add\_course\_to\_student/<filter\_name>/')

def add\_course(filter\_name) - добавление курса к ученику

@application.route('/help\_add\_course/<filter\_name>/', methods=['POST', 'GET'])

def help\_add\_course(filter\_name) - вспомогательная функция для add\_course

@application.route('/adding\_course', methods=['POST', 'GET'])

def adding\_course(): - вспомогательная функция для add course, добавляет курс к ученику

@application.route('/help\_filter/<filter\_name>/', methods=['POST', 'GET'])

def help\_lesson(filter\_name): - вспомогательная функция к create lesson

@application.route('/changing\_course', methods=['POST'])

def changing\_course() - изменение курсов

@application.route('/course\_student')

def course\_from\_student() - страница со школьными предметами

@application.route('/course\_student/<subject\_id>/')  
def course\_in(subject\_id) - курсы в школьных предметах

@application.route('/missions\_in/<lesson\_id>/view')

def view\_tasks(lesson\_id) - задачи в уроках, без конкретики

@application.route('/mission\_in/<lesson\_id>/<task\_id>/')

def task\_from\_student(lesson\_id, task\_id) - конкретная задача в уроке

def clever(task\_id): - функция для JS