@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

Загрузка объекта пользователя по его ID. Функция нужна для входа/выхода пользователя на сайт.

class User(db.Model, UserMixin):

id = db.Column(db.Integer(), primary\_key=True)

username = db.Column(db.String(length=30), nullable=False, unique=True)

email\_address = db.Column(db.String(length=50), nullable=False, unique=True)

password\_hash = db.Column(db.String(length=60), nullable=False)

budget = db.Column(db.Integer(), nullable=False, default=1000)

items = db.relationship('Item', backref='owned\_user', lazy=True)

Объект пользователя. Имеет в себе такие переменные, как: ID пользователя, имя пользователя, электронный адрес, зашифрованный пароль, сумму денег на счете и приобретенные товары.

@property

def prettier\_budget(self):

if len(str(self.budget)) >= 4:

return f'{str(self.budget)[:-3]},{str(self.budget)[-3:]}$'

else:

return f"{self.budget}$"

Функция для форматирования отображения количества денег на счете. Чтобы после 3 знаков стояла запятая.

@property

def password(self):

return self.password

Простая функция, которая возвращает пароль.

@password.setter

def password(self, plain\_text\_password):

self.password\_hash = bcrypt.generate\_password\_hash(plain\_text\_password).decode('utf-8')

Функция, которая устанавливает пароль, предварительно его шифруя с помощью алгоритма utf-8.

def check\_password\_correction(self, attempted\_password):

return bcrypt.check\_password\_hash(self.password\_hash, attempted\_password)

Функция для проверки проверки пароля.

def can\_purchase(self, item\_obj):

return self.budget >= item\_obj.price

Функция для проверки возможности купить товар. (Хватает ли на него денег).

def can\_sell(self, item\_obj):

return item\_obj in self.items

Функция продажи товара.

class Item(db.Model):

id = db.Column(db.Integer(), primary\_key=True)

name = db.Column(db.String(length=30), nullable=False, unique=True)

price = db.Column(db.Integer(), nullable=False)

category = db.Column(db.String(length=20), nullable=False, unique=True)

description = db.Column(db.String(length=1024), nullable=False, unique=True)

data = db.Column(db.LargeBinary)

owner = db.Column(db.Integer(), db.ForeignKey('user.id'))

Класс товара. Имеем такие переменные, как: ID товара, название, цену, категорию, описание, данные самого товара, ID владельца.

def buy(self, user):

self.owner = user.id

user.budget -= self.price

db.session.commit()

Функция приобретения товара. То есть, заполнения его поля владельца и уменьшения суммы денег на счете у пользователя.

def sell(self, user):

self.owner = None

user.budget += self.price

db.session.commit()

Функция продажи товара пользователем. То есть, возврат денег на счет и удаление пользователя из поля владельца у этого товара.

@app.route('/')

@app.route('/home')

def home\_page():

return render\_template('home.html')

Функция-обработчик, загружающий стартовую страницу сайта, при входе на него или по ссылке.

@app.route('/market', methods=['GET', 'POST'])

@login\_required

def market\_page():

purchase\_form = PurchaseItemForm()

selling\_form = SellItemForm()

if request.method == "POST":

#Purchase Item Logic

purchased\_item = request.form.get('purchased\_item')

p\_item\_object = Item.query.filter\_by(name=purchased\_item).first()

if p\_item\_object:

if current\_user.can\_purchase(p\_item\_object):

p\_item\_object.buy(current\_user)

flash(f"Вау! Ты забираешь {p\_item\_object.name} за {p\_item\_object.price}$", category='success')

else:

flash(f"Сори, но у тебя нет на это денег {p\_item\_object.name}!", category='danger')

#Sell Item Logic

sold\_item = request.form.get('sold\_item')

s\_item\_object = Item.query.filter\_by(name=sold\_item).first()

if s\_item\_object:

if current\_user.can\_sell(s\_item\_object):

s\_item\_object.sell(current\_user)

flash(f"Ее! Ты продал {s\_item\_object.name} назад в шоп. Найс деал!", category='success')

else:

flash(f"Что-то не так с продажей {s\_item\_object.name}", category='danger')

return redirect(url\_for('market\_page'))

if request.method == "GET":

items = Item.query.filter\_by(owner=None)

owned\_items = Item.query.filter\_by(owner=current\_user.id)

return render\_template('market.html', items=items, purchase\_form=purchase\_form, owned\_items=owned\_items, selling\_form=selling\_form)

Функция-обработчик для входа на страницу магазина. Создает формы купить/продать товар, и обрабатывает их нажатие с помощью push-уведомлений.

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

def register\_page():

form = RegisterForm()

if form.validate\_on\_submit():

user\_to\_create = User(username=form.username.data,

email\_address=form.email\_address.data,

password=form.password1.data)

db.session.add(user\_to\_create)

db.session.commit()

login\_user(user\_to\_create)

flash(f"Зафрендились! Буду звать тебя {user\_to\_create.username}", category='success')

return redirect(url\_for('market\_page'))

if form.errors != {}: #If there are not errors from the validations

for err\_msg in form.errors.values():

flash(f'Что-то не так, но не переживай, решим: {err\_msg}', category='danger')

return render\_template('register.html', form=form)

Функция-обработчик, отправляющая пользователя на страницу регистрации. И обрабатывает форму регистрации.

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

def login\_page():

form = LoginForm()

if form.validate\_on\_submit():

attempted\_user = User.query.filter\_by(username=form.username.data).first()

if attempted\_user and attempted\_user.check\_password\_correction(

attempted\_password=form.password.data

):

login\_user(attempted\_user)

flash(f'Четко! Запишу тебя как: {attempted\_user.username}', category='success')

return redirect(url\_for('market\_page'))

else:

flash('Ты что-то путаешь, или пароль, или почту', category='danger')

return render\_template('login.html', form=form)

Функция-обработчик, отправляющая пользователя на страницу входа на сайт. И обрабатывает форму входа.

@app.route('/logout')

def logout\_page():

logout\_user()

flash("Пока-пока", category='info')

return redirect(url\_for("home\_page"))

Функция-обработчик, которая совершает выход пользователя из аккаунта и возвращает его на главную страницу.

class RegisterForm(FlaskForm):

def validate\_username(self, username\_to\_check):

user = User.query.filter\_by(username=username\_to\_check.data).first()

if user:

raise ValidationError('Это имя занято, придумай что-то оригинальное')

def validate\_email\_address(self, email\_address\_to\_check):

email\_address = User.query.filter\_by(email\_address=email\_address\_to\_check.data).first()

if email\_address:

raise ValidationError('Ты что, украл чью-то почту?')

username = StringField(label='Ник Нейм:', validators=[Length(min=2, max=30), DataRequired()])

email\_address = StringField(label='эМэйл (типа Электронная почта):', validators=[Email(), DataRequired()])

password1 = PasswordField(label='Пароль:', validators=[Length(min=6), DataRequired()])

password2 = PasswordField(label='Еще раз пароль:', validators=[EqualTo('password1'), DataRequired()])

submit = SubmitField(label='Зарегать')

Класс формы регистрации. Имеет два метода для проверки правильности имени пользователя и электронного адреса. И такие переменные, как: имя пользователя, электронный адрес, два пароля, кнопку для регистрации.

class LoginForm(FlaskForm):

username = StringField(label='User Name:', validators=[DataRequired()])

password = PasswordField(label='Password:', validators=[DataRequired()])

submit = SubmitField(label='Sign in')

Класс формы входа на сайт. Имеет переменные: имя пользователя, пароль и кнопку отправки.

class PurchaseItemForm(FlaskForm):

submit = SubmitField(label='Purchase Item!')

Класс кнопки покупки товара.

class SellItemForm(FlaskForm):

submit = SubmitField(label='Sell Item!')

Класс кнопки продажи товара.