BIG BAZAR

BigBazar

MVP платформы электронной коммерции

API backend проекта

Get Started

Проект ориентирован на создание надежного и функционального решения, способного удовлетворить основные потребности пользователей в сфере электронной коммерции.

Основная цель проекта "BigBazar" - предложить рынку электронной коммерции эффективное, безопасное и удобное в использовании решение для управления пользователями и товарами. Проект разработан с акцентом на безопасность данных, интуитивно понятный интерфейс и высокую производительность системы.

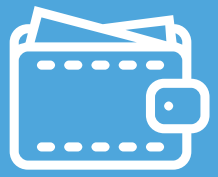
Market situation

Современный рынок электронной коммерции продолжает стремительно расти и развиваться. По прогнозам экспертов, объем мирового рынка электронной торговли к 2023 году достигнет нескольких триллионов долларов, что подчеркивает его огромный потенциал и значимость. Это ростовое пространство, однако, не лишено своих вызовов и проблем.

Основная идея ВВ состоит в том, чтобы предложить рынку решение, которое будет не только технологически продвинутым, но и легким в использовании, безопасным и масштабируемым.

Одной из ключевых задач для платформ электронной коммерции является обеспечение безопасности данных пользователей. В эпоху цифровизации информационная безопасность становится приоритетом, так как утечки данных могут нанести серьезный ущерб как пользователям, так и самим платформам.

пользователи ожидают высокого уровня удобства и легкости использования веб-сайтов и приложений для электронной торговли. Они предпочитают платформы, которые предлагают простые и интуитивно понятные интерфейсы, быстрые и удобные способы регистрации и авторизации, а также эффективное управление товарами.



Важным фактором является и технологическая адаптация. С ростом числа интернет-пользователей и увеличением объемов транзакций, платформам электронной коммерции необходимо масштабировать свои системы для обработки большего количества запросов и данных, не теряя при этом в производительности.

Task BB

Исходя из текущей ситуации на рынке электронной коммерции, проект "BigBazar" был разработан с целью предложить решение, которое удовлетворяет следующим ключевым потребностям:

01 Эффективное управление товарами

Включение функционала для удобного и управления товарами в базе данных, с возможностями добавления, редактирования и просмотра товаров, а также их активации и деактивации. Простота регистрации и авторизации обеспечивает более высокий уровень лояльности клиентов.

02 Масштабируемость и производительность

Проект должен быть способен адаптироваться к растущему числу пользователей и увеличивающимся объемам данных без потери в производительности. Реализация этой цели достигается за счет использования современных технологий и оптимизированной архитектуры.

03 Безопасность данных

В ответ на возрастающую потребность в защите персональных данных пользователей, "BigBazar" разработан с акцентом на использование передовых методов шифрования и безопасной архитектуры. Это обеспечивает надежную защиту данных клиентов и транзакций.

04 Технологическая актуальность

Применение современного стека технологий, включая Python, PostgreSQL, FastAPI, Tortoise ORM, и Pydantic, гарантирует, что "BigBazar" соответствует современным требованиям к программному обеспечению и предоставляет гибкие возможности для дальнейшего развития и интеграции.

Action BB

В рамках проекта были предприняты следующие ключевые шаги для достижения определенных целей:

Разработка Безопасной Архитектуры:

- Внедрение современных методов шифрования и аутентификации для обеспечения безопасности данных пользователей.
- Реализация системы токенов для безопасной авторизации пользователей и предотвращения несанкционированного

Реализация Системы Управления Товарами:

- Создание модулей для добавления, редактирования, просмотра и изменения статуса товаров.
- Интеграция системы с базой данных PostgreSQL для эффективного хранения и обработки информации о товарах.

Обеспечение Масштабируемости и Производительности:

- Применение асинхронного программирования в FastAPI для повышения производительности при обработке запросов.
- Использование оптимизированных запросов к базе данных и эффективного кэширования для уменьшения времени отклика.

Техническая Реализация:

- Применение Python для гибкости и мощности программирования.
- Использование Tortoise ORM для эффективной работы с базой данных.
- Применение Pydantic для строгой валидации и обработки данных, что улучшает надежность и безопасность приложения.

- В качестве основы проекта была выбрана архитектура Model-View-Controller (MVC), что обеспечивает гибкость, модульность и удобство в поддержке и масштабировании приложения.

- Язык программирования Python был выбран за его мощь и гибкость, а PostgreSQL используется как надежная и производительная система управления базами данных.

- В проекте применяется FastAPI, современный веб-фреймворк, который отличается высокой скоростью работы и поддержкой асинхронного программирования.

- Tortoise ORM, асинхронный ORM, идеально сочетается с FastAPI и обеспечивает эффективное взаимодействие с базой данных.

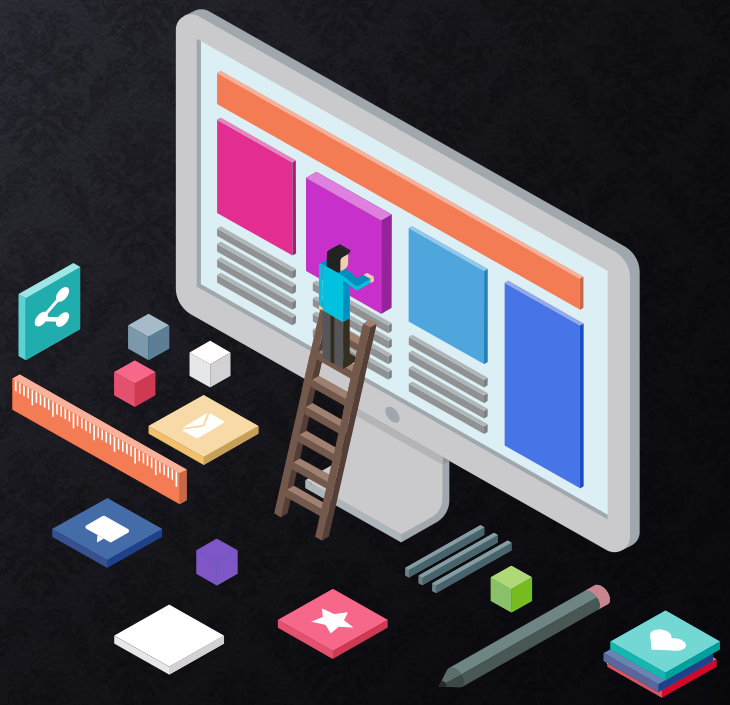
- Pydantic используется для валидации и управления данными, что гарантирует строгую типизацию и упрощает обработку данных.

Безопасность

JWT Аутентификация и шифрование паролей с Tortoise ORM на FastApi

```python
OlyaEf *
class User(models.Model):

    name = fields.CharField(max_length=150)
    email = fields.CharField(max_length=255, unique=True)
    phone = fields.CharField(max_length=15, unique=True)
    password = fields.CharField(max_length=255)
    created_at = fields.DatetimeField(auto_now_add=True)
    updated_at = fields.DatetimeField(auto_now=True)

    OlyaEf
    class PydanticMeta:
        app = 'models'
        exclude = ['password']

    2 usages  OlyaEf *
    def set_password(self, raw_password):

        raw_password = raw_password.encode('utf-8')  # Преобразование пароля в байты
        salt = bcrypt.gensalt()
        self.password = bcrypt.hashpw(raw_password, salt)

    1 usage  OlyaEf *
    def check_password(self, raw_password):

        raw_password = raw_password.encode('utf-8')  # Преобразование пароля в байты
        return bcrypt.checkpw(raw_password, self.password.encode('utf-8'))
```

```python
load_dotenv()

SECRET_KEY = os.getenv("SECRET_KEY")
ALGORITHM = os.getenv("ALGORITHM")

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")


5 usages  OlyaEf *
async def get_current_user(token: str = Depends(oauth2_scheme)) -> User:

    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )

    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        username: str = payload.get("sub")
        if username is None:
            raise credentials_exception
    except JWTError:
        raise credentials_exception

    user = await User.get_or_none(email=username)
    if user is None:
        raise credentials_exception
    return user
```



Управление и Масштабируемость

- Использование Pydantic для валидации входящих данных от пользователя при регистрации и авторизации.
- Создание моделей товаров и их управление через Tortoise ORM, обеспечивающее взаимодействие с базой данных PostgreSQL.
- Использование асинхронных запросов в FastAPI для повышения производительности

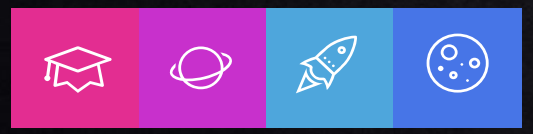
```python
class UserBase(BaseModel):

    email: EmailStr = Field( default: ..., max_length=255)
    phone: Optional[str] = None


4 usages  OlyaEf *
class UserRegistration(UserBase):

    name: str
    password: str
    confirm_password: str

    OlyaEf *
    def __setattr__(self, name, value):
        if name == 'confirm_password':
            if 'password' in self.__dict__ and value != self.__dict__['password']:
                raise ValueError("Password and confirmation password do not match")
        super().__setattr__(name, value)

    OlyaEf
    class Config:
        from_attributes = True

    Config: ClassVar[Config]  # Аннотация, указывающая, что это не поле модели
```

```python
class Product(models.Model):
    name = fields.CharField(max_length=150)
    description = fields.TextField()
    price = fields.DecimalField(max_digits=10, decimal_places=2)
    is_active = fields.BooleanField(default=False)
    created_at = fields.DatetimeField(auto_now_add=True)
    updated_at = fields.DatetimeField(auto_now=True)
    owner = fields.ForeignKeyField(model_name='models.User', relate

    OlyaEf *
    def __str__(self) -> str:
        return self.name

    OlyaEf
    class PydanticMeta:
        exclude = ['owner', 'created_at', 'updated_at', 'is_active'
```

```python
@products_router.delete( path: "/products/{product_id}", response_model=dict)
async def delete_product(product_id: int, current_user=Depends(get_current_user)):
    """
    Удаление продукта. Доступно только авторизованным пользователям.
    """
    success = await ProductService.delete_product(product_id)
    if not success:
        raise HTTPException(status_code=404, detail="Product not found")
    return {"message": "Product deleted successfully"}
```

Отчет о Покрытии Тестами (Coverage Report)

Покрытие кода тестами - это важный показатель, который помогает обеспечить стабильность и надежность нашего приложения.



Name	Stmts	Miss	Cover
bb/__init__.py	0	0	100%
bb/cart/__init__.py	0	0	100%
bb/cart/models.py	18	8	56%
bb/core/__init__.py	0	0	100%
bb/core/config.py	13	0	100%
bb/factory.py	10	0	100%
bb/main.py	8	1	88%
bb/products/__init__.py	0	0	100%
bb/products/models.py	13	1	92%
bb/products/routes.py	26	2	92%
bb/products/schemas.py	14	0	100%
bb/products/services.py	52	18	65%
bb/security/__init__.py	0	0	100%
bb/security/auth.py	23	4	83%
bb/service/__init__.py	0	0	100%
bb/service/constants.py	2	0	100%
bb/users/__init__.py	0	0	100%
bb/users/models.py	27	8	70%
bb/users/routes.py	57	9	84%
bb/users/schemas.py	41	4	90%
bb/users/services.py	60	12	80%
tests/__init__.py	0	0	100%
tests/conftest.py	45	0	100%
tests/products_test.py	57	0	100%
tests/users_test.py	53	0	100%
TOTAL	519	67	87%

BIG BAZAR

GO TO REDOC http://localhost:8000/redoc

Good luck

and Have a Nice Day

Ваш спикер Оля Ефимовских
поток 3.0. Группа Python-dev 16