УДК: 004.658
DOI 10.33514/1694-7851-2023-2-299-309

Тлебаев М.Б.
техн. илим. док., проф.
М. Х. Дулати атындагы Тараз регионалдык университети
Казакстан, Тараз ш.
Шайнусов Р.Т.
магистрант
М.Х. Дулати атындагы Тараз регионалдык университети
Казакстан, Тараз ш.
Бексултанов Ж.Т.
физ.-мат. илим. канд., доц.
И. Арабаев атындагы Кыргыз мамлекеттик университети
Бишкек ш.
jenish1979@mail.ru
Ачекеев К.С.
ага окутуучу
Бишкек ш.
И. Арабаев атындагы Кыргыз мамлекеттик университети
kadyrbekachekeev@mail.ru

DJANGO ORM ПЛАТФОРМАСЫНДА БИЗНЕСТЕ МОДЕЛДЕРДИ ТҮЗҮҮ ЖАНА МААЛЫМАТ БАЗАСЫН КУРУУ ӨЗГӨЧӨЛҮКТӨРҮ

Аннотация. Макалада бизнестин негизги аспектилерин, анын ичинде компанияларды, менчик түрлөрүн, бизнес түрлөрүн жана каржылык көрсөткүчтөрдү чагылдырган моделдерди түзүү каралат. Бара-бара маалымат базасынын структурасы иштелип чыгат, Foreign Key кандай моделдерди өз ара байланыштыруу үчүн мамилелерди кантип туура колдонууну көрсөтөт. Ар кандай талаа түрлөрү, моделдердин ортосундагы байланыштар жана ишенимдүү жана структураланган маалымат базасын түзүүгө жардам бере турган маалыматтарды текшерүү жолдору сүрөттөлөт. Моделдерди иштеп чыгуунун мыкты тажрыйбалары дагы каралат, маалымат класстары сүрөттөлөт жана Django ORM платформанын маалымат базасы менен оптималдуу иштөө үчүн мүмкүнчүлүктөрү колдонулат.

Негизги сөздөр: маалымат моделдерин түзүү, маалымат базасы, бизнес, Django ORM, долбоорлоо, моделдер, компаниялар, менчик түрлөрү, бизнес түрлөрү, каржы.

Тлебаев М.Б.
док. техн. наук, проф.
Таразский региональный университет имени М.Х. Дулати
Казахстан, г. Тараз
Шайнусов Р.Т.
магистрант
Таразский региональный университет имени М.Х. Дулати
Казахстан, г. Тараз
Бексултанов Ж.Т.
канд. физ.-мат. наук, доц.
Кыргызский государственный университет имени И. Арабаева
г. Бишкек

jenish1979@mail.ru
Ачекеев К.С.
старший преподаватель
Кыргызский государственный университет имени И. Арабаева
г. Бишкек
kadyrbekachekeev@mail.ru

ОСОБЕННОСТИ ФОРМИРОВАНИЯ МОДЕЛЕЙ И ПОСТРОЕНИЕ БАЗЫ ДАННЫХ В БИЗНЕСЕ НА ПЛАТФОРМЕ DJANGO ORM

Аннотация. В статье рассматривается создание моделей, отражающих основные аспекты бизнеса, включая компании, типы собственности, типы бизнеса и финансовые показатели. Постепенно разрабатывается структура базы данных, демонстрируя, как правильно использовать отношения Foreign Key для связи различных моделей между собой. Описываются различные типы полей, связи между моделями и способы валидации данных, которые помогут нам создать надежную и структурированную базу данных. Также рассматриваются лучшие практики проектирования моделей, описываются классы данных и используются возможности платформы Django ORM для оптимальной работы с базой данных.

Ключевые слова: формирование моделей данных, база данных, бизнес, Django ORM, проектирование, модели, компании, типы собственности, типы бизнеса, финансовые показатели.

Tlebaev M.B.
Doctor of Technical Sciences, Professor
Taraz Regional University named after M.Kh.Dulati
Kazakhstan, Taraz c.
Shainusov R.T.
Master's Student
Taraz Regional University named after. M.Kh. Dulati
Kazakhstan, Taraz c.
Beksultanov Zh.T.
Candidate of Physical and Mathematical Sciences, Associate Professor
Kyrgyz State University named after I. Arabaev
Bishkek
jenish1979@mail.ru
Achekeev K. S.
Senior Lecturer
Kyrgyz State University named after I. Arabaeva
Bishkek c.
kadyrbekachekeev@mail.ru

FEATURES OF FORMING MODELS AND BUILDING A DATABASE IN BUSINESS ON THE DJANGO ORM PLATFORM

Abstract. The article discusses the creation of models reflecting the main aspects of business, including companies, types of ownership, types of business and financial indicators. The database structure is gradually being developed, demonstrating how to properly use Foreign Key relationships to link different models to each other. It describes various types of fields, relationships between models, and data validation methods that will help us create a reliable and structured database. The best prac-

tices of model design are also considered, data classes are described and the capabilities of the Django ORM platform are used for optimal database work..

Keywords: formation of data models, database, business, django ORM, design, models, companies, types of ownership, types of business, financial indicators.

В современном бизнесе база данных является неотъемлемой частью успешной работы. Правильное проектирование моделей данных играет ключевую роль в эффективной организации и управлении информацией о компаниях. В данной статье мы рассмотрим процесс формирования моделей данных для построения базы данных бизнеса с использованием Django ORM.

Для реализации базы данных можно предложить следующие модели: модель данных разработана на основе фреймворка Django. Она включает следующие модели:

BaseModel. Базовая абстрактная модель, которая содержит поля **created** и **updated**. Она используется в качестве основы для всех моделей в данной базе данных.

OwnershipType. Модель, которая содержит информацию о типах собственности бизнеса. Она включает поля **name** и **description**.

BusinessType. Модель, которая содержит информацию о типах бизнеса. Она включает поля **name** и **description**.

Company. Модель, которая содержит информацию о компаниях. Она включает следующие поля:

- **name**: название компании;
- **user**: связь с пользователем, создавшим компанию;
- **tax_id**: БИН или ИИН компании (должен содержать 12 цифр);
- **slug**: URL-адрес компании;
- **ownership_type**: тип собственности компании;
- **business_type**: тип бизнеса компании;
- **description**: описание деятельности компании;
- **address**: адрес компании;
- **phone_number**: номер телефона компании;
- **email**: электронный адрес компании;
- **website**: веб-сайт компании;
- **image**: логотип или фото компании.

Employee

Модель, которая содержит информацию о сотрудниках компаний. Она включает следующие поля:

- **company**: связь с компанией, в которой работает сотрудник;
- **first_name**: имя сотрудника;
- **last_name**: фамилия сотрудника;
- **middle_name**: отчество сотрудника (необязательно);
- **position**: должность сотрудника.

Все поля моделей включают соответствующие валидаторы и/или ограничения на длину или уникальность. Например, поле **tax_id** в модели **Company** должно содержать ровно 12 цифр и быть уникальным в базе данных. Кроме того, в модели **Company** определены связи с моделями **OwnershipType** и **BusinessType**, которые обеспечивают связь между типами собственности и типами бизнеса компаний.

Класс **Financials** содержит поля, которые хранят финансовые показатели компании за определенный год.

- **company** – связь с моделью **Company**, указывает на компанию, чьи финансовые данные хранятся в этой записи.
- **year** – год, за который представлены данные.

- **revenue** – общая выручка компании за указанный год.
- **expenses** – расходы, связанные с бизнесом компании за указанный год.
- **gross_profit** – валовая прибыль компании за указанный год.
- **net_income** – чистая прибыль компании за указанный год.
- **taxes** – налоги, заплаченные компанией за указанный год.
- **dividends** – дивиденды, выплаченные компанией за указанный год.
- **margin** – маржа компании за указанный год.

Модель **Financials** позволяет хранить в базе данных финансовые данные о компаниях за различные годы. Эти данные могут быть полезны при анализе финансового состояния компании и позволяют в дальнейшем проводить аналитику и прогнозирование ее дальнейшего развития[4].

Модель данных в целом является хорошо структурированной и позволяет эффективно хранить и управлять информацией о субъектах малого и среднего бизнеса.

Рассмотрим каждую модель по отдельности.

class BaseModel(models.Model):
created = models.DateTimeField(auto_now_add=**True**)
updated = models.DateTimeField(auto_now=**True**)
class Meta:
abstract = **True**

BaseModel – это абстрактный класс модели Django, который содержит два поля: created и updated. Эти поля автоматически заполняются при создании и обновлении объекта соответственно. Таким образом, BaseModel может использоваться в качестве базовой модели для всех других моделей, которые должны иметь поля created и updated.

class OwnershipType(BaseModel):
name = models.CharField(**«Название формы собственности»**, max_length=255)
description = models.TextField(**«Описание формы собственности»**, blank=**True**)
def __str__(self):
return self.name

Модель OwnershipType используется для хранения информации о типе собственности бизнеса. Она содержит следующие поля:

- name – название формы собственности;
- description - описание формы собственности (необязательно).

class BusinessType(BaseModel):
name = models.CharField(**«Название типа бизнеса»**, max_length=255)
description = models.TextField(**«Описание типа бизнеса»**, blank=**True**)
def __str__(self):
return self.name

Модель BusinessType используется для хранения информации о типе бизнеса. Она содержит следующие поля:

- name - название типа бизнеса;
- description - описание типа бизнеса (необязательно).

 # Регулярно выражение для валидатора поля tax_id
BIN_IIN_REGEX = **«^\d{12}$»** *# Только 12 цифр*
class Company(BaseModel):
««««Модель для хранения информации о компании»»»

name = models.CharField(
«Название компании»,

```
        max_length=200,
        unique=True,
        help_text=«Название бизнеса должно быть уникальным и не использоваться другими компаниями в вашей юрисдикции.»,)
        # user
        user = models.ForeignKey(
        settings.AUTH_USER_MODEL, related_name=«companies», on_delete=models.PROTECT)
        # БИН & ИИН
        tax_id = models.CharField(
        «Введите ИИН или БИН»,
        max_length=12,
        validators=[RegexValidator(regex=BIN_IIN_REGEX, message=«Введите 12-й номер»)],
        unique=True,
        help_text=«Введите ИИН если вы индивидуальный предприниматель, БИН в другом случае.»,)
    slug = models.SlugField(max_length=200, unique=True, blank=True)
        # форма собственности
        ownership_type = models.ForeignKey(
        OwnershipType,
        on_delete=models.PROTECT,
        related_name=«companies»,
        verbose_name=«Форма собственности»,
        help_text=«Выберите форму собственности, которой принадлежит компания.»,)
        # тип бизнеса
        business_type = models.ForeignKey(
        BusinessType,
        on_delete=models.PROTECT,
        related_name=«companies»,
        verbose_name=«Тип бизнеса»,
        help_text=«Выберите тип бизнеса, которым занимается компания.»,)
        description = models.TextField(
        «Описание деятельности компании», blank=True, help_text=«Необязательно.»)
        address = models.CharField(«Адрес компании», max_length=200)
        phone_number = PhoneNumberField(«Номер телефона компании», region=«KZ»)
        email = models.EmailField(«Электронный адрес компании»)
        website = models.URLField(
        «Веб-сайт компании», blank=True, help_text=«Необязательно.»)
        # Фото, логотип компании
        image = models.ImageField(
        «Логотип или фото компании»,
        upload_to=«company_images/»,
        blank=True,
        help_text=«Необязательно»,)
        def save(self, *args, **kwargs):
        # Если slug не установлен (первое создание объекта)
        if not self.slug:
        # slug
        self.slug = slugify(unidecode(self.name))
        return super().save(*args, **kwargs)
```

Класс **Company** является моделью, которая хранит информацию о компании. Он содержит следующие поля:

- **name** – название компании, обязательное поле с максимальной длиной 200 символов. Уникальное поле, которое не может быть использовано другими компаниями в вашей юрисдикции.
- **user** – связь с пользователем, который является владельцем компании. Это обязательное поле.
- **tax_id** – ИНН (индивидуальный номер налогоплательщика) или БИН (бизнес идентификационный номер) компании. Уникальное поле, которое необходимо для идентификации компании. Длина поля должна быть строго 12 символов. Валидируется с регулярным выражением, используемое для проверки того, что введенный текст состоит из ровно 12 цифр.
- **slug** – уникальный идентификатор компании для использования в URL. Это автоматически генерируемое поле, которое основано на названии компании.
- **ownership_type** – связь с моделью **OwnershipType**, которая представляет форму собственности компании. Это обязательное поле.
- **business_type** – связь с моделью **BusinessType**, которая представляет тип бизнеса компании. Это обязательное поле.
- **description** – описание деятельности компании. Это необязательное поле, которое позволяет пользователю описать свою компанию.
- **address** – адрес компании. Это обязательное поле с максимальной длиной 200 символов.
- **phone_number** – номер телефона компании. Это обязательное поле. Он хранится в формате международного номера телефона, используя **django-phonenumber-field.**
- **email** – адрес электронной почты компании. Это обязательное поле.
- **website** - веб-сайт компании. Это необязательное поле.
- **image** – логотип или фото компании. Это необязательное поле, которое хранит изображения в формате **ImageField.**
- **created** и **updated** – автоматически добавляемые поля, которые хранят дату и время создания и обновления объекта.

Класс **Company** представляет собой один из основных компонентов базы данных, которая предназначена для хранения информации о компаниях. Он имеет связь с другими моделями, такими как **OwnershipType** и **BusinessType**, что позволяет пользователям более детально описывать тип собственности и тип бизнеса.

Данный код содержит переопределение метода **save** в модели **Company**. При сохранении объекта **Company**, если поле **slug** не заполнено (то есть это первое создание объекта), метод **save** вычисляет значение для **slug** на основе поля **name**, используя функцию **slugify** и **unidecode**. Затем метод **save** вызывает реализацию родительского класса **super().save(*args, **kwargs)**, чтобы выполнить стандартное сохранение объекта[10].

```
# Choices для пола
GENDER_CHOICES = [
(«M», «Мужской»),
(«F», «Женский»),]
class Employee(BaseModel):
««««Модель для хранения информации о сотрудниках компании»««
 # Связь с компанией
company = models.ForeignKey(
Company, on_delete=models.CASCADE, related_name=«employees»)
# Информация о сотруднике
```

```
first_name = models.CharField(«Имя», max_length=255)
last_name = models.CharField(«Фамилия», max_length=255)
middle_name = models.CharField(
«Отчество», max_length=255, blank=True, help_text=«Необязательно» )
position = models.CharField(«Должность», max_length=255)
 # Контактная информация
 email = models.EmailField(«Электронный адрес»)
phone_number = PhoneNumberField(«Номер телефона», region=«KZ»)
 # Доп. Информация
 date_of_birth = models.DateField(
«Дата рождения», blank=True, null=True, help_text=«Необязательно»)
image = models.ImageField(
«Фото сотрудника»,
upload_to=«employee_images/»,
 blank=True,
 help_text=«Необязательно»,)
gender = models.CharField(
«Пол»,
 max_length=1,
 choices=GENDER_CHOICES,
 blank=True,
 help_text=«Необязательно»,)
def __str__(self):
middle_name = f» {self.middle_name}» if self.middle_name else ««
return f»{self.last_name} {self.first_name}» + middle_name
```

Конечная модель Employee представляет собой модель базы данных для хранения информации о сотрудниках компании. Она содержит следующие поля:

- company: ForeignKey связь с моделью Company, чтобы установить связь с компанией, к которой относится сотрудник.
- first_name: CharField для имени сотрудника, максимальная длина 255 символов.
- last_name: CharField для фамилии сотрудника, максимальная длина 255 символов.
- middle_name: CharField для отчества сотрудника, максимальная длина 255 символов. Это поле не является обязательным.
- position: CharField для должности сотрудника, максимальная длина 255 символов.
- email: EmailField для электронной почты сотрудника.
- phone_number: PhoneNumberField для номера телефона сотрудника. В качестве региона задана Казахстанская телефонная зона (region=«KZ»).

Кроме того, модель Employee также имеет поля, которые определены в базовой абстрактной модели BaseModel, такие как created и updated для хранения даты и времени создания и обновления объекта.

```
YEARS = [(year, year) for year in range(2000, timezone.now().year)]
class Financials(BaseModel):
««««Neutral and universal model for financial information about a company»»««
 company = models.ForeignKey(Company, on_delete=models.CASCADE)
year = models.IntegerField(
verbose_name=«Отчет за год»,
choices=YEARS,
 help_text=«За какой год информация ниже актуальна», )
revenue = models.DecimalField(
```

```
    max_digits=10,
    decimal_places=2,
    blank=True,
    null=True,
    verbose_name=«Общая выручка»,
    help_text=«Показывает общую сумму денег, которую компания заработала за определенный год»,  )
    expenses = models.DecimalField(
    max_digits=10,
    decimal_places=2,
    blank=True,
    null=True,
    verbose_name=«Расходы»,
    help_text=«Поле может включать все расходы, связанные с бизнесом, такие как зарплата сотрудников, аренда помещения, закупка оборудования и т.д.», )
    gross_profit = models.DecimalField(
    max_digits=10,
    decimal_places=2,
    blank=True,
    null=True,
    verbose_name=«Валовая прибыль»,
    help_text=«Это поле показывает разницу между выручкой и себестоимостью товаров или услуг.», )
    net_income = models.DecimalField(
    max_digits=10,
    decimal_places=2,
    blank=True,
    null=True,
    verbose_name=«Чистая прибыль»,
    help_text=«Это поле может показывать сумму денег, которую компания заработала после вычета всех расходов.»,)
    taxes = models.DecimalField(
    max_digits=10,
    decimal_places=2,
    blank=True,
    null=True,
    verbose_name=«Налоги»,
    help_text=«Это поле может включать информацию о том, сколько налогов компания заплатила за определенный период времени.»,)
    dividends = models.DecimalField(
    max_digits=10,
    decimal_places=2,
    blank=True,
    null=True,
    verbose_name=«Дивиденды»,
    help_text=«Это поле может показывать сумму денег, которую компания выплатила своим акционерам»,)
    margin = models.DecimalField(
```

max_digits=10,
decimal_places=2,
blank=**True**,
null=**True**,
verbose_name=«**Маржа**»,
help_text=«**Это поле показывает процентную прибыль компании, которая остается после вычета всех расходов от общей выручки.**»,)
def __str__(self):
return f'Финансовый показатели субъекта «{self.company.name**}»'**

Модель «Financials» представляет собой нейтральную и универсальную модель для финансовой информации о компании. В этой модели есть следующие поля:

- «company» – поле типа ForeignKey, которое связывает модель «Financials» с моделью «Company».
- «year» – поле типа IntegerField с выбором значений, представленных списком YEARS, которые в данном случае означают годы с 2000 по текущий год.
- «revenue» – поле типа DecimalField, которое показывает общую сумму денег, которую компания заработала за определенный год.
- «expenses» – поле типа DecimalField, которое может включать все расходы, связанные с бизнесом, такие как зарплата сотрудников, аренда помещения, закупка оборудования и т.д.
- «gross_profit» – поле типа DecimalField, которое показывает разницу между выручкой и себестоимостью товаров или услуг.
- «net_income» – поле типа DecimalField, которое может показывать сумму денег, которую компания заработала после вычета всех расходов.
- «taxes» – поле типа DecimalField, которое может включать информацию о том, сколько налогов компания заплатила за определенный период времени.
- «dividends» – поле типа DecimalField, которое может показывать сумму денег, которую компания выплатила своим акционерам.
- «margin» – поле типа DecimalField, которое показывает процентную прибыль компании, которая остается после вычета всех расходов от общей выручки.

Также в модели есть метод «**str**», который возвращает строку, содержащую имя компании и информацию о финансовых показателях.

@admin.register(Company)
class CompanyAdmin(admin.ModelAdmin):
prepopulated_fields = {**«slug»**: (**«name»**,)}
admin.site.register(BusinessType)
admin.site.register(OwnershipType)
admin.site.register(Employee)
admin.site.register(Financials)

Эти строки кода связаны с административной панелью Django, которая предоставляет интерфейс для управления данными вашей модели веб-приложения.

Первая строка @admin.register(Company) регистрирует модель Company в административной панели Django. Декоратор @admin.register предоставляет более простой способ зарегистрировать модель в административной панели Django, чем метод admin.site.register.

class CompanyAdmin(admin.ModelAdmin): определяет, как модель Company будет отображаться и какие действия будут доступны в административной панели Django. Например, prepopulated_fields говорит Django, что поле slug будет заполняться автоматически на основе значения поля name.

admin.site.register регистрирует другие модели (BusinessType, OwnershipType, Employee) в административной панели Django без дополнительной настройки.

Эти строки кода помогают нам управлять данными в административной панели Django, позволяя вам добавлять, редактировать и удалять данные с помощью простого пользовательского интерфейса. На рисунке 1 показана административная панель, а на рисунке 2 форма добавления организации.

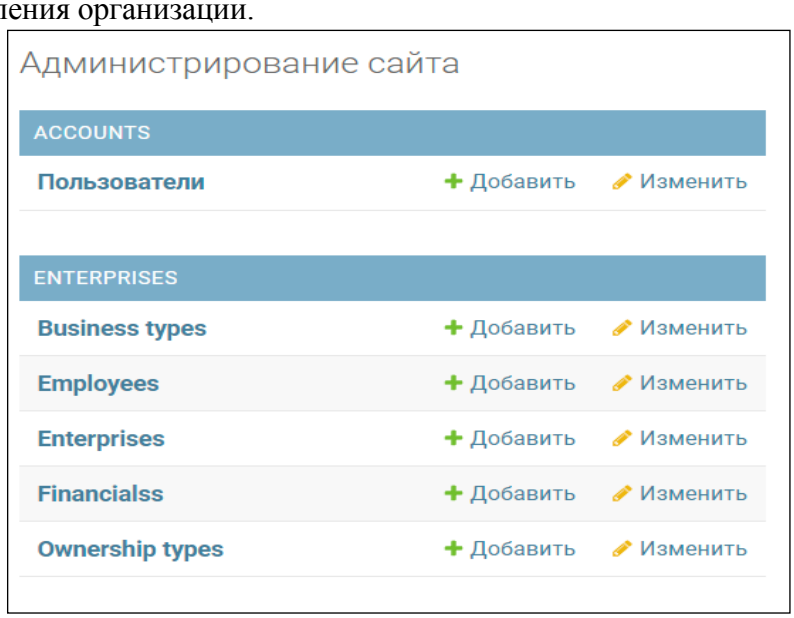

Рисунок 1. Административная панель Django

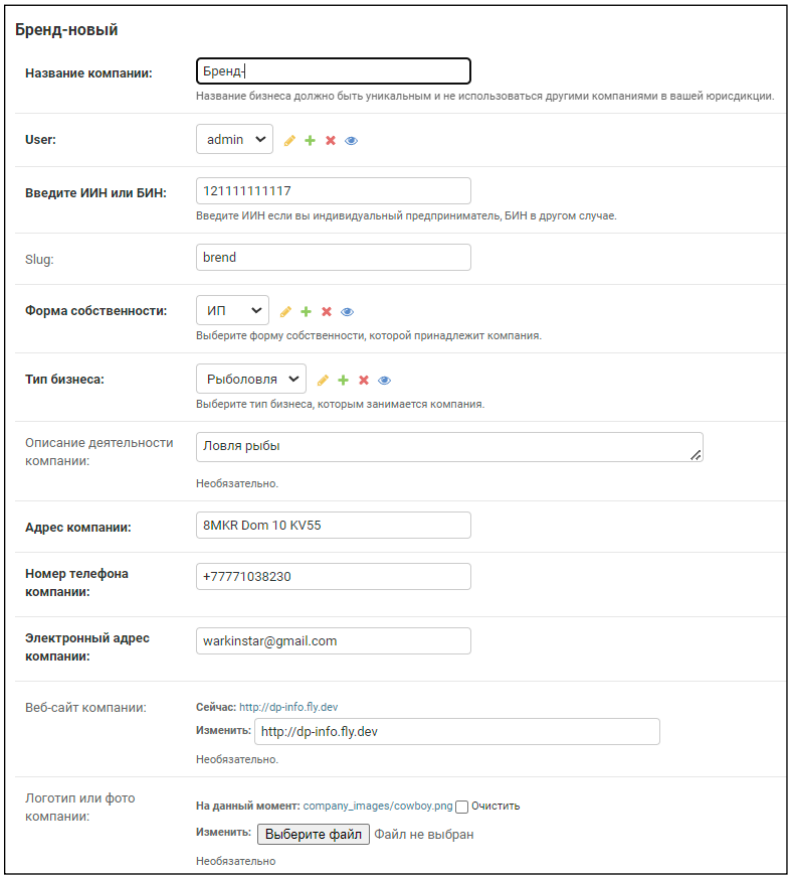

Рисунок 2. Форма добавления предприятия в административной панели

В этой статье мы рассмотрели процесс формирования моделей данных для построения базы данных бизнеса в Django ORM. Мы изучили основные модели, такие как модель компании, типы собственности и бизнеса, а также модель финансовых показателей.

Мы обратили внимание на различные типы полей, связи между моделями и способы валидации данных, которые помогут нам создать надежную и структурированную базу данных. Также мы рассмотрели лучшие практики проектирования моделей и использования возможностей Django ORM для оптимальной работы с базой данных.

При разработке базы данных бизнеса важно учесть особенности вашей организации и ее потребности. Грамотное проектирование моделей данных поможет упростить работу с информацией о компаниях, их финансовых показателях и других важных аспектах бизнеса.

Мы надеемся, что данная статья помогла вам получить полезные знания и инсайты в процессе формирования моделей данных для базы данных бизнеса в Django ORM. Благодаря Django ORM вы сможете эффективно управлять информацией о компаниях и строить надежные и масштабируемые бизнес-приложения.

Применение правильных моделей данных и использование Django ORM позволят вам сосредоточиться на развитии вашего бизнеса, минимизировать ошибки и упростить процессы работы с базой данных[7].

Мы желаем вам успехов в проектировании и разработке базы данных для вашего бизнеса, и надеемся, что данная статья станет полезным руководством в вашем путешествии по созданию надежных и эффективных баз данных[13].

Список использованной литературы:

1. Бизнес-план малого и среднего предприятия: составление и оценка эффективности / Панков А.Ф. – М.: РХХ, 2019.

2. Организация малого и среднего бизнеса / Шульц Л. – М.: Омега-Л, 2018.

3. Экономика малого и среднего бизнеса / Алексеев А.Г. – М.: ЮНИТИ-ДАНА, 2017.

4. Анализ и управление финансами малого и среднего бизнеса / Лучко Н. – СПб.: Питер, 2018.

5. Интернет-маркетинг для малого бизнеса / Глушаков М. – М.: Инфра-М, 2016.

6. Маркетинг малого и среднего бизнеса / Хусаинов А. – М.: Издательский дом «Деловой квартал», 2019.

7. Управление проектами малого и среднего бизнеса / Петров В. – М.: Бизнес-школа «Интел-Синтез», 2017.

8. Закон Республики Казахстан от 4 июля 2013 года № 129-V «О Национальной палате предпринимателей Республики Казахстан.

9. «What is Django?»., https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

10. B.C. Винсент., «Django для начинающих», 2019 г.

11. A. Меле «Django by example», 2015 г.

12. https://articlekz.com/article/15401

13. https://gender.stat.gov.kz/ru/category/6

Рецензент: канд. техн. наук, и.о. доц. Керимов У.Т.