**МиНобрнауки россии**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В.Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Институт *Энергетики, информационных технологий и управляющих систем*

Кафедра *Программного обеспечения вычислительной техники и автоматизированных систем*

Направление подготовки *09.03.04 Программная инженерия*

(шифр, наименование)

Направленность (профиль) образовательной программы *Разработка программно-информационных систем*

(наименование)

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

на тему:

***Разработка программной системы автоматизации бюджетных процессов в области энергосбережения***

**Студент (ка) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Харитонов Сергей Дмитриевич

**Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**канд. техн. наук, доц. Поляков В.М.

**Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**ст.пр. Буханов Д. Г.

**К защите допустить**

**Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/** *Поляков В.М .***/**

**«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.**

**Белгород 2020 г.**

**МиНобрнауки россии**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В.Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Институт  *Энергетики, информационных технологий и управляющих систем*

Кафедра *Программного обеспечения вычислительной техники и автоматизированных систем*

Направление подготовки *09.03.04 Программная инженерия*

(шифр, наименование)

Направленность (профиль) образовательной программы *Разработка программно-информационных систем*

(наименование)

|  |  |
| --- | --- |
|  | Утверждаю:  Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г. |

**ЗАДАНИЕ**

на выпускную квалификационную работу студента (ки)

*Харитонова Сергея Дмитриевича*

(Фамилия Имя Отчество)

1. Вид выпускной квалификационной работы (ВКР) *бакалаврская работа*

2. Тема ВКР *РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ АВТОМАТИЗАЦИИ БЮДЖЕТНЫХ ПРОЦЕССОВ В ОБЛАСТИ ЭНЕРГОСБЕРЕЖЕНИЯ*

утверждено приказом по университету от *«27» января 2020 г. № 2/47*

3. Срок сдачи студентом законченной ВКР *22.06.2020*

4. Исходные данные *учёт, автоматизация учёта производства, Java Spring, PostgreSQL, Java Persistence API, Web API, REST, SOAP, Data Transfer Object*

5. Содержание ВКР (перечень подлежащих разработке разделов)

*Производственный учёт кондитерских изделий;*

*Проектирование серверной части программного обеспечения учёта производства*

*кондитерских изделий;*

*Разработка серверной части программного обеспечения учёта производства кондитерских*

*изделий.*

6. Перечень графического материала *Титульный слайд, Цель и задачи, Актуальность работы,*

*Учёт производства на АО «КФ «Белогорье», Программные аналоги, Общая архитектура приложения, Средства разработки, Схема взаимодействия сервисов приложения, База данных «Цех», Диаграмма классов (model) «Цех», Диаграмма классов (controller) «Цех», Основные*

*запросы к системе, Основные запросы к системе, Итоги*

Консультанты по работе с указанием относящихся к ним разделов

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Консультант | Задание выдал  (подпись, дата) | Задание принял  (подпись, дата) |

Дата выдачи задания «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Задание принял к исполнению

(подпись студента)

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов работы | Срок выполнения  этапов работы | Примечание |
| **1** | Постановка задачи, описание предметной области, изучение процессов учёта производства АО «Кондитерская фабрика «Белогорье» | *27.01.2020 – 16.03.2020* | *Выполнено* |
| **2** | Прoектирoвaние серверной части программного обеспечения учёта производства кондитерских изделий | *16.03.2020 – 13.04.2020* | *Выполнено* |
| **3** | Разработка и тестирование программного обеспечения учёта производства кондитерских изделий | *14.05.2020 – 15.06.2020* | *Выполнено* |
| **4** | Оформление пояснительной записки, презентации | *16.06.2020 – 19.06.2020* | *Выполнено* |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Харитонов Сергей Дмитриевич

(подпись) (Ф.И.О.)

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Буханов Дмитрий Геннадьевич.

(подпись)

**«Результаты проверки ЭВ ВКР на заимствование»**

Кафедра Программного обеспечения вычислительной техники и автоматизированных систем

Студент Харитонов С.Д. ПВ-42

Фамилия Инициалы Группа Дата

Тема ВКР Разработка серверной части программного обеспечения

учёта производства кондитерских изделий

ВКР прошла проверку на объем заимствований.

Итоговая оценка оригинальности:\_\_\_\_\_\_\_\_\_\_\_

Работу проверил Гончарук А.А.

Подпись Дата Фамилия Инициалы

Руководитель

ВКР доцент, канд. техн. наук Буханов Д.Г.

Должность, уч. степень, звание Подпись Дата Фамилия Инициалы

**OПРЕДЕЛЕНИЯ, СOКРAЩЕНИЯ И OБOЗНAЧЕНИЯ**

ПЭС – Программа ЭнергоСбережения

РСО – РесурсоСнабжающая Компания

СФПЭС – Система Формирования Паспорта ЭнергоСбережения

СOДЕРЖAние

[ВВЕДЕНИЕ 8](#_Toc99929156)

[1. АНАЛИЗ СОСТОЯНИЯ ИССЛЕДОВАНИЯ В ОБЛАСТИ ИСПОЛЬЗОВАНИЯ ЭНЕРГЕТИЧЕСКИХ РЕСУРСОВ 9](#_Toc99929157)

[1.1. Энергоресурсы и их виды 9](#_Toc99929158)

[1.2. Бюджетные процессы в сфере энергосбережения 10](#_Toc99929159)

[1.3. Программа энергосбережения 11](#_Toc99929160)

[1.4. Анализ теплотехнических характеристик для разработки ПЭС 12](#_Toc99929161)

[1.4.1. Общие данные об организации 12](#_Toc99929162)

[1.4.2. Здания на балансе организации 13](#_Toc99929163)

[1.4.3. Энергопотребление ресурсов 14](#_Toc99929164)

[1.4.4. Сведения об оснащенности приборами учета 15](#_Toc99929165)

[1.5. Анализ существующих подходов подбора энергосберегающих мероприятий 16](#_Toc99929166)

[1.5.1. Анализ методики расчета потенциал энергосбережения и целевые уровни потребления энергоресурсов 17](#_Toc99929167)

[1.5.2. Выбор энергосберегающих мероприятий 18](#_Toc99929168)

[1.6. Постановка задачи 19](#_Toc99929169)

[2. РАЗРАБОТКА СФПЭС 8](#_Toc99929170)

[2.1. Разработка общей структуры СФПЭС 8](#_Toc99929171)

[2.1.1. Разработка подсистема авторизации 10](#_Toc99929172)

[2.1.2. Разработка подсистемы хранения данных об организации 11](#_Toc99929173)

[2.1.3. Разработка подхода для валидации данных об организации на основе ИНС. 13](#_Toc99929174)

[*Структура ИНС* 14](#_Toc99929175)

[*Обучающая выборка и ее параметры* 15](#_Toc99929176)

[2.1.4. Разработка подсистемы формирования ПЭС 16](#_Toc99929177)

[2.1.5. Разработка подсистемы подбора энергосберегающих мероприятий 18](#_Toc99929178)

[2.1.6. Разработка подсистемы оплаты 19](#_Toc99929179)

[2.2. Разработка архитектуры СФПЭС 19](#_Toc99929180)

[2.2.1. Разработка архитектуры серверной части 21](#_Toc99929181)

[2.2.2. Разработка общей структуры клиентского приложения 23](#_Toc99929182)

[2.3. Формирование API приложения 25](#_Toc99929183)

[3. РЕАЛИЗАЦИЯ СФПЭС 8](#_Toc99929184)

[3.1. Реализация подсистемы регистрации, авторизации и аутентификации 9](#_Toc99929185)

[3.1.1. Реализация модуля разграничения доступа 10](#_Toc99929186)

[3.1.2. Реализация модуля API 11](#_Toc99929187)

[3.1.3. Реализация разделения доступа к API приложения 13](#_Toc99929188)

[3.2. Реализация подсистемы хранения данных об учреждении 14](#_Toc99929189)

[3.2.1. Реализация модуля моделей 14](#_Toc99929190)

[3.2.2. Реализация модуля API 15](#_Toc99929191)

[3.3. Реализация подсистемы формирования паспорта ПЭС 17](#_Toc99929192)

[3.3.1. Реализация модуля получения данных из подсистемы хранения информации об организации 19](#_Toc99929193)

[3.3.2. Реализация модуля агрегирования данных для разделов 20](#_Toc99929194)

[3.3.3. Реализация модуля заполнения шаблона паспорта ПЭС 20](#_Toc99929195)

[3.3.4. Реализация модуля API 21](#_Toc99929196)

[3.4. Реализация подсистемы оплаты 21](#_Toc99929197)

[3.4.1. Реализация модуля хранения платежных реквизитов 23](#_Toc99929198)

[3.4.2. Реализация модуля взаимодействия с банковской системой 24](#_Toc99929199)

[3.4.3. Реализация модуля API 25](#_Toc99929200)

[3.5. Алгоритм подбора энергосберегающих мероприятий 26](#_Toc99929201)

[3.6. Реализация подсистемы валидации данных об организации на основе MLP классификатора 27](#_Toc99929202)

[3.6.1. Реализация MLP классификатора 27](#_Toc99929203)

[3.6.2. Алгоритм валидации данных об организации на основе MLP классификатора 28](#_Toc99929204)

[3.6.3. Результаты работы подсистемы 29](#_Toc99929205)

[3.7. Реализация клиента СФПЭС 30](#_Toc99929206)

[3.7.1. Роутинг 30](#_Toc99929207)

[3.7.2. Компоненты и контейнеры 31](#_Toc99929208)

[3.7.3. Взаимодействие с API 32](#_Toc99929209)

[4. АППРОБАЦИЯ И ТЕСТИРОВАНИЕ СФПЭС 8](#_Toc99929210)

[4.1. Система автоматического тестирования 8](#_Toc99929211)

[4.1.1. Модульное тестирование 8](#_Toc99929212)

[4.1.2. Интеграционное тестирование 8](#_Toc99929213)

[4.2. Представление графа интерфейса 8](#_Toc99929214)

[4.3. Анализ технических характеристик работы ПО 8](#_Toc99929215)

[4.3.1. Технические характеристики 8](#_Toc99929216)

[4.3.2. Статистика работы системы за 3 месяца 8](#_Toc99929217)

[ЗAКЛЮЧЕНИЕ 9](#_Toc99929218)

[СПИСOК ЛИТЕРAТУРЫ 10](#_Toc99929219)

[ПРИЛOЖЕНИЕ А 13](#_Toc99929220)

# ВВЕДЕНИЕ

# АНАЛИЗ СОСТОЯНИЯ ИССЛЕДОВАНИЯ В ОБЛАСТИ ИСПОЛЬЗОВАНИЯ ЭНЕРГЕТИЧЕСКИХ РЕСУРСОВ

В современном мире значительную роль в быту и ежедневной рутине играют энергоресурсы. Трудно представить кухню без газовой плиты, жилую комнату без электрического освещения, а дом без системы отопления. Для доставки энергоресурсов конечным потребителям существует огромная электроэнергетическая система.

Бюджетные организации, как и любые другие, так же потребляют различные энергетические ресурсы.

## Энергоресурсы и их виды

Учреждения бюджетного сектора используют и потребляют следующие энергетические ресурсы:

* тепловая энергия;
* природный газ;
* электрическая энергия;
* холодная водоснабжение;
* горячая водоснабжение.

Для доставки энергетических ресурсов конечным потребителям существует обширная инфраструктура. Потребители, производители и организации осуществляющие доставку энергетических ресурсов объединяются в энергетические системы на основе вида ресурса, с которым они взаимодействуют.

Энергетическая система представляет из себя совокупность объектов энергетики и энергопринимающих установок потребителей энергии, связанных общим режимом работы в едином технологическом процессе производства, передачи и потребления электрической̆ энергии в условиях централизованного оперативно-диспетчерского управления в электроэнергетике. [https://www.elec.ru/viewer?url=/files/2021/09/27/GOST-21027-2021.pdf]

Данные системы имеют значительное влияние на работу экономики страны и благосостояние ее граждан, именно поэтому они нуждаются в государственном регулировании. Согласно энергетической стратегии России одной из главных задач государственного регулирования в сфере энергетики является повышение энергетической и экологической эффективности российской экономики и энергетики, в том числе за счет структурных изменений и активизации технологического энергосбережения.[ <https://centerpolit.org/national-security/energeticheskaya-strategiya-rossii-na-period-do-2030-goda/>]

## Бюджетные процессы в сфере энергосбережения

Для контроля объектов бюджетного сектора в области потребления энергоресурсов используются следующие бюджетные процессы:

* расчет потенциала энергосбережения и целевых уровней снижения потребления энергоресурсов; [https://energy.midural.ru/wp-content/uploads/2020/09/PMER\_15\_07\_2020\_425.pdf]
* представление деклараций о потреблении энергетических ресурсов; [https://set.rk.gov.ru/uploads/txteditor/set/attachments/d4/1d/8c/d98f00b204e9800998ecf8427e/phpjQqoEY\_707.pdf]
* разработка и контроль исполнение программ энергосбережения.[https://www.consultant.ru/document/cons\_doc\_LAW\_93978/]

В данной работе будут рассматриваться бюджетные процессы связанные с разработкой программ энергосбережения и их паспортов.

## Программа энергосбережения

Одним из инструментов повышения эффективности использования энергоресурсов является разработка и реализация программ энергосбережения.

Программы энергосбережения, разрабатываемые в ходе ведения бюджетных процессов, являются инструментом повышения эффективности использования энергоресурсов и сокращения их потребления в стоимостном и материальном выражениях.

Согласно, федеральному закону «Об энергосбережении и о повышении энергетической эффективности и о внесении изменений в отдельные законодательные акты Российской Федерации» каждая бюджетная организация обязана разрабатывать и актуализировать ПЭС. Также помимо бюджетных организаций ПЭС разрабатывают муниципалитеты и субъекты РФ. [https://www.consultant.ru/document/cons\_doc\_LAW\_93978/]

Целью ПЭС является сокращение потребления используемых энергоресурсов, а также полное использование потенциала энергосбережения организации. Для этого на основе данных об организации и потребляемых энергоресурсах специалистом в сфере энергосбережения составляется план внедрения мероприятий, по энергосбережению и повышению энергетической эффективности, на ближайшие 3–5 лет. В ходе исполнения плана проводятся различные мероприятия: от исключительно технических (установка приборов учета) до организационных (проведение инструктажей и обучение персонала), нацеленные на более эффективное использование энергоресурсов.

Так же в ходе разработки ПЭС формируется паспорт программы энергосбережения и ежегодная промежуточная отчетность.

Паспорт ПЭС содержит информацию об организации, ее имуществе (здания, транспорт, приборы учета и другое) и план внедрения энергосберегающих мероприятий. Исходя из плана внедрения мероприятий и потребления энергоресурсов за предшествующие года вычисляется плановое ежегодное потребление, которого организация должна достичь в ходе реализации ПЭС.

Ежегодная промежуточная отчетность должна содержать информацию о реализации энергосберегающих мероприятий и сравнение планового потребление энергоресурсов с фактическим. На основании промежуточной отчетности ПЭС может корректироваться для лучшего достижения энергосберегающего эффекта.

## Анализ теплотехнических характеристик для разработки ПЭС

Разработка ПЭС представляет из себя объемное исследование организации(энергоаудит) на предмет возможностей улучшения показателей энергоэффективности. Для этого специалисту в сфере энергосбережения предоставляется такая информация об организации как:

* общая информация об организации;
* информация о транспорте, принадлежащем организации или арендуемом ей;
* технические характеристики зданий и строений организации;
* данные о потреблении энергоресурсов за предшествующие года;
* сведения об оснащенности приборами учета.

Все перечисленные блоки информации необходимы для корректной оценки потенциала энергосбережения, установки целевых показателей и составления максимально эффективного плана внедрения энергосберегающих мероприятий.

## Общие данные об организации

Общие данные об организации содержат юридическую информацию об организации:

* наименование;
* юридический адрес;
* информация о должностных лицах (директоре, лице ответственном за энергосбережение);
* ИНН;
* код ОГРН;
* код ОКВЕД;
* и другое.

Данный блок информации не несет практической пользы для разработки ПЭС т. к. не содержит теплотехнических характеристик организации. Но он необходим лицу, разрабатывающему паспорт ПЭС, т. к. согласно нормативно правовым актам паспорт ПЭС должен содержать юридическую информацию об организации, для которой разрабатывается ПЭС.

## Здания на балансе организации

Энергетические ресурсы потребляются в рамках зданий и помещений используемых организацией, их теплотехнические характеристики используются для разработки плана внедрения энергосберегающих мероприятий и для расчета целевых показателей ПЭС.

Важным критерием плана внедрения энергосберегающих мероприятий является его реализуемость и целесообразность: не имеет смысла включать в план мероприятия по замене деревянных стеклопакетов, если в здании в принципе нет таковых.

Поэтому для разработки плана внедрения энергосберегающих мероприятий необходимо учитывать технические характеристики зданий, чтобы исключить бессмысленные мероприятия и наоборот понять какие мероприятия принесут максимальную пользу.

Также такие характеристики как отапливая площадь и среднее количество посетителей в сутки используются для расчета удельных показателей потребления энергоресурсов, таких как Гкал/м2 (тепловая энергия), м3/человека(вода) и других. А исходя из удельных показателей рассчитывается потенциал экономии и целевые уровни снижения потребления энергоресурсов, необходимые для разработки ПЭС.

Используют следующие технические характеристики зданий:

* среднее число посетителей в сутки;
* общая площадь;
* отапливаемая площадь;
* количество этажей;
* материал наружных стен;
* наличие утепления фасада;
* площадь остекления;
* количество вводов энергоресурсов, а также информация о том какое количество вводов оборудовано приборами учета.

## Энергопотребление ресурсов

Основной целью ПЭС является снижение объема потребления энергоресурсов. Для корректной установки данного целевого показателя требуется учитывать множество факторов, в том числе и объем энергоресурсов, потребляемый организацией до внедрения ПЭС.

Плановое потребление энергоресурсов рассчитывается исходя из базового потребления и плана реализации энергосберегающих мероприятий.

За базовое потребление принимают потребление энергоресурсов в базовом году (предшествующий первому году начала реализации ПЭС).

Для расчета планового потребления предполагается, что без внедрения мероприятий по энергосбережению объем потребляемых энергоресурсов был бы не изменен и равнялся базовому потреблению. Исходя из этого плановое потребление принимается как разница потребления в базовом году и экономии, полученной вследствие реализации энергосберегающих мероприятий.

Так же в паспорте ПЭС для отслеживания динамики изменения потребления энергоресурсов указывается потребление за два года предшествующие базовому.

Следовательно, для разработки ПЭС необходимо знать потребление энергоресурсов за базовый год и 2 года предшествующих ему.

Объем потребления энергоресурсов указывается для следующих видов энергоресурсов:

* тепловая энергия;
* природный газ;
* электрическая энергия;
* холодное водоснабжение;
* горячее водоснабжение.

## Сведения об оснащенности приборами учета

Для составления ПЭС важна достоверность информации об объеме потребляемых энергетических ресурсов.

Для точной оценки объема потребления используются приборы коммерческого учета. Прибор (или узел) учета тепловой энергии — это комплекс приборов и устройств, обеспечивающих учет тепловой энергии, массы (объема) теплоносителя, а также контроль и регистрацию его параметров. Как правило они устанавливаются на границу ответственности между РСО и абонентом. [<http://www.energosovet.ru/bul_stat.php?idd=118>]

Одним из наиболее эффективных энергосберегающих мероприятий является установка прибора учета, которое в среднем позволяет сэкономить до 15% потребляемого объема. Сама установка прибора учета не уменьшает объем потребление, но позволяет отказаться от расчета по средним показателям, который зачастую превышает реальное потребление.

Так же в паспорт ПЭС вносится информация не только о количестве вводов, оборудованных приборами учета, но и информация по каждому конкретному прибору учета. Поэтому для разработки ПЭС необходима следующая информация:

* место установки;
* тип энергоресурса, подлежащий учету;
* класс точности;
* заводской номер;
* год ввода в эксплуатацию;
* дата последней поверки;
* меж поверочный интервал.

## Анализ существующих подходов подбора энергосберегающих мероприятий

План по внедрению энергосберегающих мероприятий является ядром ПЭС и представляет из себя список энергосберегающих мероприятий, сроки их проведения, а также количество и источники бюджетных средств, выделяемых на их реализацию.

План составляется лицом, имеющим высокую квалификацию в сфере энергосбережения, которое использует технические характеристики организации, целевые уровни снижения потребления энергоресурсов и потенциал энергосбережения, для выбора максимально эффективных и реализуемых в рамках организации мероприятий.

Данный процесс представляет из широкое поле для внедрения автоматизации, ведь именно он является наиболее затратным этапом разработки ПЭС, как в материальном, так и во временном плане.

Для автоматизации и контроля данного процесса необходимо реализовать два этапа:

* расчет целевых уровней снижения и потенциала энергосбережения;
* подбор энергосберегающих мероприятий на основе входных данных.

## Анализ методики расчета потенциал энергосбережения и целевые уровни потребления энергоресурсов

При составлении плана внедрения энергосберегающих мероприятий необходимо соблюсти баланс между размером предполагаемой экономии и ее целесообразностью. Для того что бы подобрать оптимальное количество мероприятий, проводится анализ организации и рассчитываются целевые уровни потребления энергоресурсов и потенциал энергосбережения.

Данные характеристики рассчитываются на основании методики из приказа 425 Министерства Энергетики.

Целевой уровень снижения потребления ресурса - плановый̆ удельный̆ годовой̆ расход ресурса, до которого государственное (муниципальное) учреждение обязано снизить свой фактический удельный годовой расход данного ресурса после его приведения к сопоставимым условиям.

Потенциал снижения потребления ресурсов рекомендуется определять как разницу между текущим уровнем потребления ресурсов государственным (муниципальным) учреждением и уровнем, при котором потребление ресурсов осуществляется максимально эффективно и соответствует высокому классу энергетической эффективности. [https://energy.midural.ru/wp-content/uploads/2020/09/PMER\_15\_07\_2020\_425.pdf]

Для определения данных показателей потребление организации за базовый год переводится в удельные показатели, так же потребление тепловой энергии дополнительно приводится к сопоставимым условиям исходя из длительности отопительного периода (формула 1.3.1) и этажности зданий (формула 1.3.2).

ЗАМЕНИТЬ НА НОРМАЛЬНУЮ ФОРМУЛУ И ПОДПИСАТЬ НОМЕР

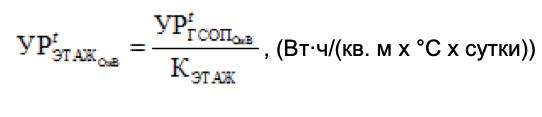
Изображение выглядит как текст

Автоматически созданное описание(1.3.1)

УРоив – удельный годовой расход тепловой энергии на нужды отопления и вентиляции в календарном году (Гкал/кв.м)

ГСОП – число градус суток отопительного периода за этот же календарный год

ЗАМЕНИТЬ НА НОРМАЛЬНУЮ ФОРМУЛУ И ПОДПИСАТЬ НОМЕР

(1.3.2)

УРгсоп – удельный годовой расход тепловой энергии на нужды отопления и вентиляции в году приведенный к сопостовимым климатическим условиям (Вт\*ч/(кв.м х С х сутки)

Кэтаж – корректировочный коэффициент на этажность и режим работы.

На основе удельных показателей и типа организации в соответствии со справочными таблицами из методики находится потенциал энергосбережения и целевой уровень потребления энергоресурсов.

Исходя из рассчитанных показателей мероприятия подбираются таким образом чтобы достичь целевого уровня снижения потребления энергоресурсов, а также полностью использовать потенциал энергосбережения в ходе реализации ПЭС.

Обычно ПЭС реализуется в течение трех лет и, согласно методике, рекомендуется разделять рассчитанное снижение потребления энергоресурсов по 50%, 25% и 25% на каждый год соответственно.

## Выбор энергосберегающих мероприятий

Для достижения целевых показателей ПЭС лицо ответственное за энергосбережение составляет план внедрения энергосберегающих мероприятий.

Энергосберегающие мероприятия можно поделить на две группы:

* нацеленные на сокращение потребления конкретных ресурсов (установка приборов учета, утепление фасадов);
* Организационные мероприятия применяемые к персоналу организации (разработка Положения о порядке стимулирования работников за экономию энергии и энергоресурсов)

После расчета целевых уровней потребления энергоресурсов, а также потенциала энергосбережения, лицо разрабатывающее ПЭС выбирает энергосберегающие мероприятия так, чтобы максимально приблизиться к целевым показателям и при этом сохранить реализуемость плана по внедрению энергосберегающих мероприятий.

Составление плана внедрения энергосберегающих мероприятий является творческим процессом, но можно выделить некоторые закономерности, которые будут использоваться для автоматизации составления плана:

* все вводы энергоресурсов, не оборудованные приборами учета, оборудуются ими;
* в случае наличия не энергоэффективных приборов освещения, они заменяются на энергоэффективные;
* не утепленные помещения, в которых работает отопление, утепляются;
* деревянные стеклопакеты заменяются на энергоэффективные;
* и т. д.

По такому принципу все неэнергоэффективное оборудование заменяется на энергоэффективное.

## Постановка задачи

Проведенный анализ существующего процесса составления ПЭС позволяет выявить следующие недостатки:

разработчик ПЭС должен иметь высокую квалификацию в сфере энергосбережения;

разработка и контроль ПЭС производится раз в три года, что снижает оперативность реагирования;

изменения в законодательстве либо других нормативных документах в среде бюджетных учреждений распространяются не равномерно;

необходим обширный объем информации (технические характеристики зданий и приборов учета, информация об объемах потребления энергоресурсов и т.д.), указанной пользователем;

отсутствие методик подбора мероприятий.

Таким образом, *целью* выпускной квалификационной работы является повышение энергоэффективности бюджетных организаций, за счет автоматизации бюджетных процессов и разработки автоматизированной системы создания и контроля программ энергосбережения и их паспортов.

**Вывод**

Подводя итоги текущей главы, можно отметить, что разработка ПЭС и ее паспорта является сложным бюджетным процессом, таящим в себе массу нюансов, расчетов и решений, принимаемых специалистом в сфере энергосбережения. Так же были выделены основные этапы разработки ПЭС и необходимые входные данные, на основе которых можно сделать вывод о наличии возможности внедрения автоматизации и ее выгоды в данном бюджетном процессе.

Самым удобным для пользователя способом автоматизации было принято использование приложения с WEB-базированным интерфейсом, позволяющего разработать паспорт ПЭС.

## РАЗРАБОТКА СФПЭС

Для разработки СФПЭС необходимо выполнить следующие этапы:

* разработать общую структуру системы и выделить системы, соответствующие функциональности;
* разработать общую архитектуру системы, а так же архитектуру клиентской и серверной частей;
* сформировать API для взаимодействия подсистем СФПЭС.

## Разработка общей структуры СФПЭС

Разработана функциональная диаграмма взаимодействия пользователя с СФПЭС(рисунок NN).

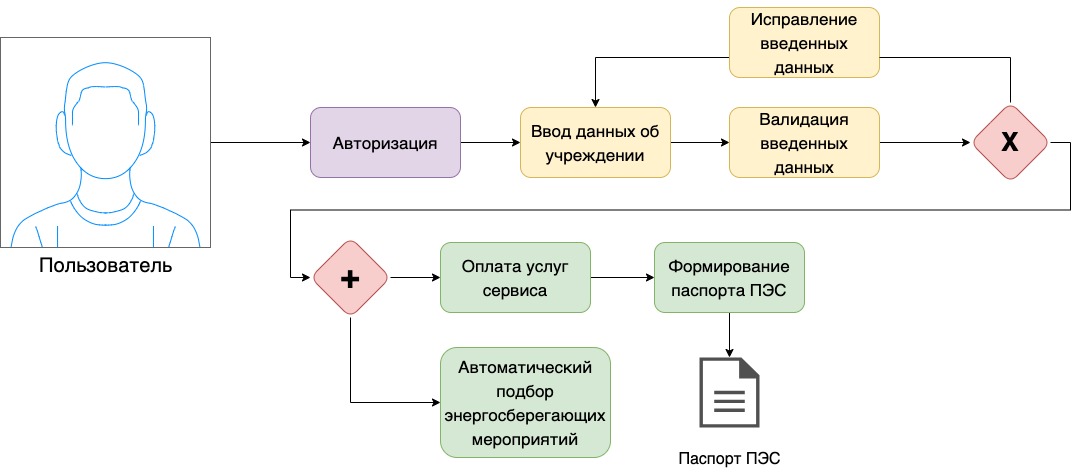


Рисунок N.N – Функциональная диаграмма взаимодействия пользователя с сервисом

На диаграмме видно 3 блока отвечающие за различные виды функциональности приложения:

* фиолетовый – функция авторизации пользователя;
* желтый – функция хранения и валидации данных об организации;
* зеленый – функция формирования ПЭС.

Для соблюдения принципа единой ответственности и сохранения гибкости системы, функция хранения и валидации данных и функция формирования ПЭС, в соответствии с диаграммой, были разбиты на меньшие части выполняющие одну конкретную задачу.

Что бы предоставить пользователю описанную функциональность, необходимо разработать подсистемы СФПЭС, реализующие конкретные функции. (Разработана)

В ходе разработки общей структуры СФПЭС были выделены следующие подсистемы:

* подсистема регистрации, авторизации и аутентификации;
* подсистема ввода данных об организации;
* подсистема валидации данных об организации;
* подсистема формирования паспорта ПЭС;
* подсистема оплаты;
* подсистема подбора энергосберегающих мероприятий.

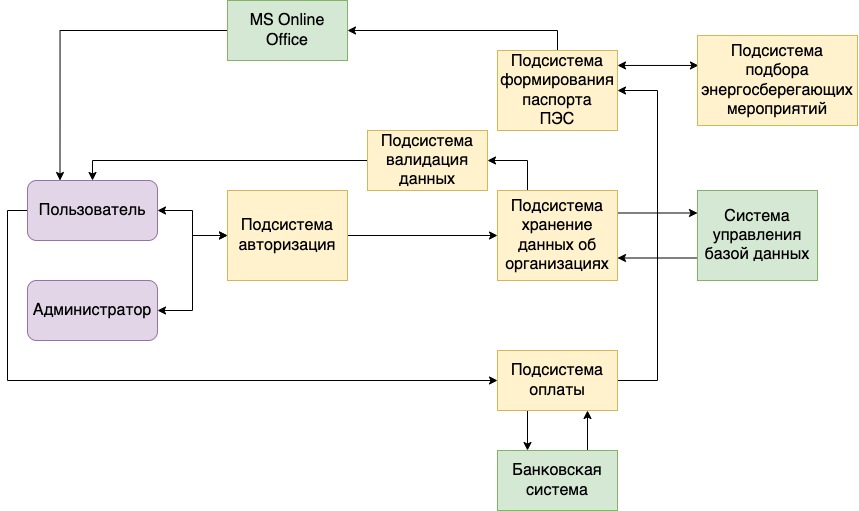


Рисунок N.N – Общая структура СФПЭС.

На рисунке N.N изображена общая структура желтым цветом отмечены описанные подсистемы, представляющие из себя ядро СФПЭС, а зеленым внешние сервисы, используемые подсистемами для реализации своей функциональности.

## Разработка подсистема авторизации

Разрабатываемая система подразумевает ввод данных об организации пользователями, вследствие этого необходимо проводить аутентификацию и авторизацию пользователей. Это необходимо для разграничения доступа пользователей к хранимым на сервере данным и исключения возможности изменить или узнать информацию о других пользователях.

СФПЭС будет распространяться как приложение с WEB-базированным интерфейсом, поэтому подсистема авторизации должна также предоставлять пользователям возможность регистрации в сервисе для последующей авторизации и аутентификации.

Так же для получения доступа к полной версии паспорта ПЭС необходимо будет произвести оплату, это так же требует реализации авторизации и аутентификации от системы для сохранения платежных данных пользователя и недопущения их утечки.

Для случая потери пароля от учетной записи пользователя необходимо предусмотреть надежный и безопасный механизм восстановления пароля.

В результате подсистема должна предоставлять API для:

* регистрации;
* авторизации-аутентификации;
* восстановления пароля.

## Разработка подсистемы хранения данных об организации

Для разработки паспорта ПЭС необходимо хранить большое количество информации об организации, в связи с тем, что использование этой информации требуется в большинстве подсистем СФПЭС, было принято решение выделить отдельную подсистему ответственную за хранение данных об организации.

Данная подсистема представляет из себя прослойку между системой управления базой данных и подсистемами разрабатываемой СФПЭС.

Для разработки подсистемы необходимо на основе анализа предметной области выделить сущности базы данных и разработать структуру, которая в дальнейшем будет реализована с помощью СУБД.

В результате анализа предметной области были выделены следующие сущности:

Организация – организация, для которой необходимо составить ПЭС у организации есть тип организации, уровень бюджета, управляющий состав (менеджеры) директор, директор в сфере энергосбережения, лицо ответственное за энергосбережение, адрес, ИНН, ОКВЭД, ОГРН, номер телефона, полное название организации, веб-сайт, ОКОГУ, номер энергетического паспорта, сведение о наличие утвержденной ПЭС, организация использует ресурсы, так же на его балансе состоят здания.

Адрес – адрес состоит из почтового индекса, региона, района, населенного пункта, улицы, дома, корпуса.

Менеджер – представляет из себя человека, у него есть фамилия, имя, отчество, должность, номер телефона, электронная почта.

Здание – у здания есть название, описание, местонахождение, площадь, количество этажей, объем. Так же остекление, отопление, информация о праве собственности, стены, чердак и подвал определенного типа, в здании могут быть установлены приборы учета.

Энергетический ресурс – энергетический ресурс, используемый организацией, для его учета может быть установлен прибор учета, он имеет определенный тип, так же хранится информация о том какая РСО его поставляет и о контракте на его поставку.

Тип ресурса. Энергетический ресурс может быть пяти видов Тепло, Электричество, Холодная вода, Горячая вода, Природный газ

Потребление энергетического ресурса - потребление энергетического ресурса как фактическое, так и материальное за определенный год

Прибор учета, устанавливаемый в здании для учета энергетического ресурса, у него есть марка, заводской номер, класс точности, год ввода в эксплуатацию, год последней поверки.

Энергосберегающее мероприятие. Для определенной организации проводятся мероприятия у них есть тип, год выполнения, затраты на внедрение и процентная экономия.

Тип энергосберегающего мероприятия - содержит описание мероприятия, стандартную экономию в процентах и по какому типу ресурса будет экономия.

Транспорт на балансе организации. Организации может принадлежать автотранспорт для него хранится информация о его названии, количестве, грузоподъемности, пробеге, удельный расход топлива норматив, удельный расход топлива факт.

Так же было выделены сущности хранящие справочную информацию такую как типы утепления, остекления, теплоснабжения и другие.

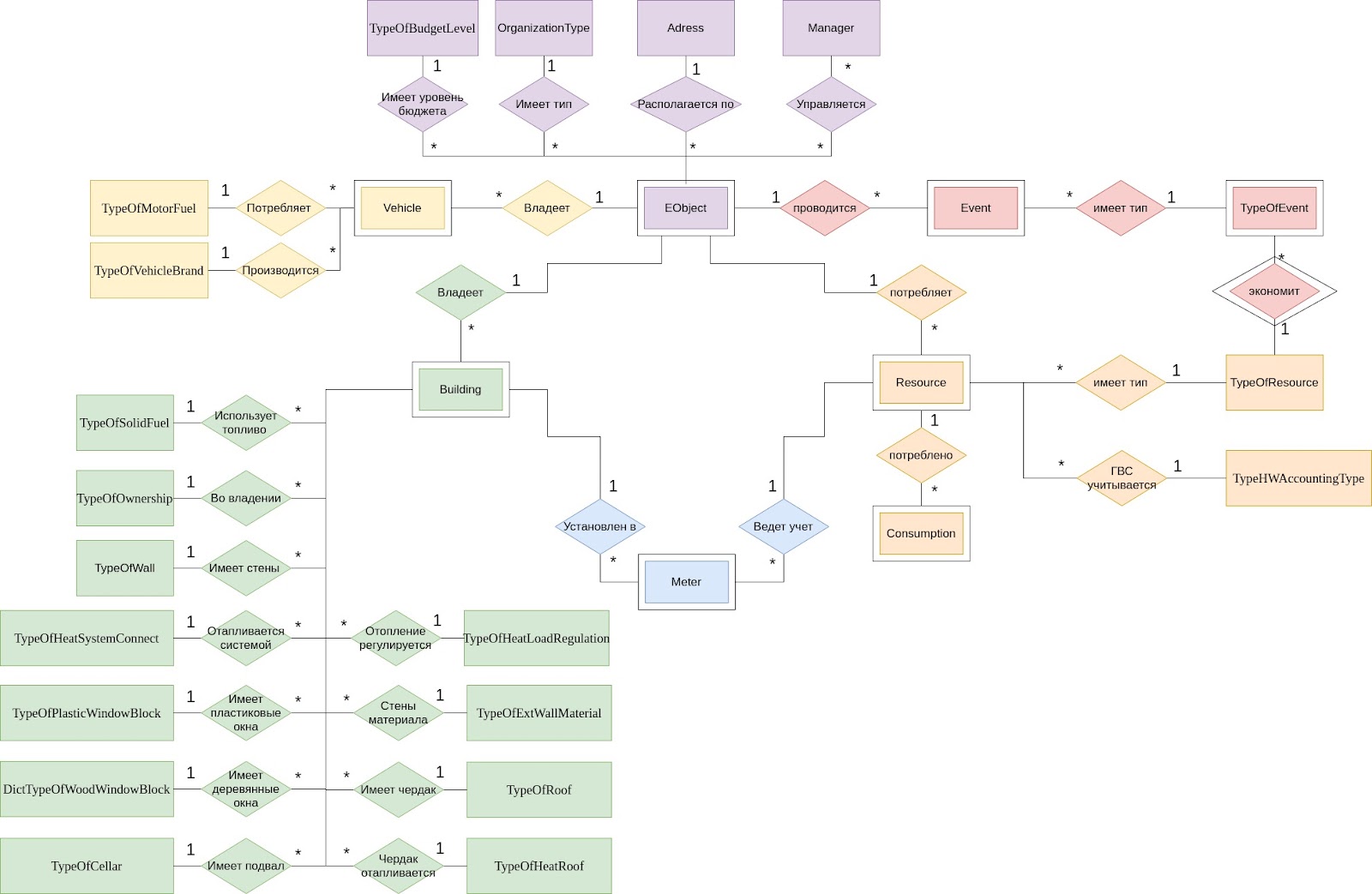


Рисунок N.N – Диаграмма сущность связь структуры базы данных

БД соответствует первой нормальной форме, так как все входящие в отношение атрибуты являются атомарными (неделимыми).

БД соответствует второй нормальной форме, так как находится в первой нормальной форме и каждый не ключевой атрибут отношения функционально зависит от первичного ключа.

БД соответствует третьей нормальной форме, так как находится во второй нормальной форме, и каждый не ключевой атрибут не транзитивно зависит от первичного ключа.

## Разработка подхода для валидации данных об организации на основе ИНС.

Основной сложностью при разработке ПЭС является получение актуальных и верных сведений об учреждении. Современные способы решения данной проблемы предполагают привлечение специалиста в области энергетики и энергосбережения для проверки, введенных пользователем, данных. Для полной автоматизации процесса было решено заменить человека на интеллектуальные средства валидации полученных данных.

Наиболее развитыми методами современного интеллектуального анализа являются методы, основанные на:

ДОБАВИТЬ ССЫЛКИ ИЗ СТАТЬИ ЗА МЕСТО ТЕХ ЧТО В СПИСКЕ

* нечеткой логике [4-6];
* деревьях решений [7-9];
* кластеризации [10, 11];
* искусственных нейронных сетях (ИНС) [8, 12-14].

Для решения поставленной задачи было принято решение использовать ИНС.

## *Структура ИНС*

Разработанная модель многослойного персептрона состоит из плотных слоев (Dense):

1.Входного слоя (dense\_1\_input) с 28-ью нейронами.

2.Слоя dense\_1, содержащего функцию активации нейронов elu (28 входов, 56 выходов).

3.Слоя dense\_2, содержащего функцию активации нейронов elu (56 входов, 112 выходов).

4.Выходного слоя dense\_3 с функцией активации sigmoid, содержащего 112 входов и 1 выход.

Сигмовидная функцию активации в слое dense\_3 используется для бинарной классификации, для малых значений функция возвращает значение, близкое к нулю, а для больших значений результат функции приближается к 1.

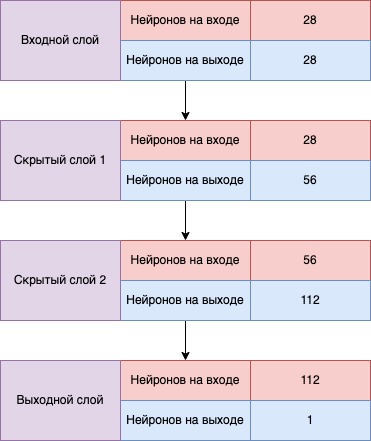


Рисунок N.N – Модель многослойного персептрона, выполняющего бинарную классификацию.

Текст (Разработанная модель классицикатора позволяет …, ее требуется обученить для обучения)

## *Обучающая выборка и ее параметры*

В качестве обучающей выборки, подаваемой на вход в разработанный классификатор, были выбраны данные организаций, которые были вручную проверены специалистом в области энергосбережения. Размер обучающей выборки составил 1912 образов, проверенных и утвержденных как корректные и 300 образов, которые были отмечены как некорректные. В качестве полей для валидации были выделены поля из таблицы 2.1.

Таблица N.N – Параметры набора данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Тип данных** | **Смысловое содержание** |
| 1 | Целое без знаковое | Количество вводов по холодной воде |
| 2 | Целое без знаковое | Количество вводов по холодной воде, оборудованных узлами коммерческого учета |
| 3 | Целое без знаковое | Количество вводов по горячей воде |
| 4 | Целое без знаковое | Количество вводов по горячей воде, оборудованных узлами коммерческого учета |
| 5 | Целое без знаковое | Количество вводов тепловой энергии |
| 6 | Целое без знаковое | Количество вводов тепловой энергии, оборудованных узлами коммерческого учета |
| 7 | Целое беззнаковое | Количество вводов электроэнергии |
| 8 | Целое беззнаковое | Количество вводов электрической энергии, оборудованных узлами коммерческого учета |
| 9 | Целое беззнаковое | Количество вводов по газу |
| 10 | Целое беззнаковое | Количество вводов по газу, оборудованных узлами коммерческого учета |
| 11 | Целое беззнаковое | Количество всех сотрудников |
| 12 | Целое беззнаковое | Общая отапливаемая площадь |
| 13 | Вещественное | Общий отапливаемый объем |
| 14 | Вещественное | Полезная площадь |
| 15 | Вещественное | Общая площадь ограждающих конструкций, включая окна |
| 16 | Вещественное | Площадь чердачного перекрытия |
| 17 | Вещественное | Площадь подвальных перекрытий |
| 18 | Целое беззнаковое | Среднее количество людей |
| 19 | Вещественное | Общая площадь |
| 20 | Вещественное | Длина здания |
| 21 | Вещественное | Ширина здания |
| 22 | Вещественное | Высота здания |
| 23 | Вещественное | Периметр здания |
| 24 | Целое беззнаковое | Этажность |
| 25 | Вещественное | Общий объем |
| 26 | Вещественное | Площадь остекления |
| 27 | Вещественное | Остекление энергосберегающими стеклопакетами (% от общего остекления) |
| 28 | Целое беззнаковое | Количество окон |

## Разработка подсистемы формирования ПЭС

Конечным продуктом работы СФПЭС является .doc файл, содержащий паспорт ПЭС. Для наполнения паспорта ПЭС информацией используются данные полученные из подсистемы хранения данных.

Поскольку для хранения и использования данных применяются разные форматы целесообразно использовать прослойку для их модификации и агрегации. Следовательно подсистема формирования ПЭС должна реализовывать две функции:

* получение и модификация данных из подсистемы хранения данных;
* формирование печатной формы паспорта ПЭС.

Для удобства модификации данных было принято решение разделить ее на два слоя получение информации из подсистемы хранения данных и их агрегация для удобства формирования печатной формы.



Рисунок N.N – Общая структура подсистемы формирования паспорта ПЭС.

Из диаграммы виден процесс формирования печатной формы. Классы типа Получатель используют данные из подсистемы хранения данных и передают их в классы типа Агрегатор, предназначенные для соединения полученных данных в соответствии с разделами паспорта ПЭС.

После получения данных в нужном формате, подсистема помещает их в шаблон паспорта ПЭС и предоставляет пользователю готовую печатную форму.

## Разработка подсистемы подбора энергосберегающих мероприятий

Для упрощения разработки плана внедрения энергосберегающих мероприятий была разработана подсистема, позволяющая на основе данных об организации порекомендовать типы энергосберегающих мероприятий, которые принесут максимальную пользу в рамках организации.

В качестве входных параметров подсистема получает данные об организации, а результатом работы подсистемы являются созданные сущности, хранящие информацию об энергосберегающих мероприятиях.

В силу того, что расчет фактической стоимости и выбор года внедрения невозможны исходя из имеющихся в системе данных эти поля остаются пустыми и в дальнейшем заполняются пользователем на основе стоимости проведения мероприятий на базе учреждения и бюджетных средств доступных организации.

Для рекомендации энергосберегающих мероприятий используются эвристики, выведенные в ходе анализа предметной области и уже разработанных ПЭС. Зачастую внедряемые энергосберегающие мероприятия в различных организациях схожи между собой, что позволит реализовать алгоритм, результатом которого будет план внедрения энергосберегающих мероприятий, практически не требующий корректировок со стороны пользователя.

## Разработка подсистемы оплаты

Для монетизации сервиса было принято решение предоставлять пользователю только фрагмент паспорта ПЭС до оплаты услуги.

Подсистема оплаты должна предоставлять интерфейс для удобной оплаты пользователем услуги предоставляемой СФПЭС. На выбор пользователю представляется два тарифа:

* оплата как физическое лицо;
* оплата как юридическое лицо.

Для оплаты в роли физического лица, необходимо реализовать возможность оплаты с помощью банковской карты, данная функция может быть реализована при помощи внешних систем и их API, предоставляемых банками

Для оплаты в роли юридического лица, по реквизитам пользователя необходимо составить договор о предоставлении услуги, после чего пользователь заверяет его печатью и получает результат оказания услуги в виде паспорта ПЭС.

Так же подсистема оплаты, должна предоставлять подсистеме формирования ПЭС информацию о том оплатил ли пользователь полную версию паспорта ПЭС. Данная информация предоставляется для принятия решения о том какой вариант паспорта получит пользователь (полный или укороченный).

## Разработка архитектуры СФПЭС

Для системы была разработана многоуровневая архитектура, состоящая из четырех уровней: клиент, API, сервер, база данных. (рисунок 2.1)

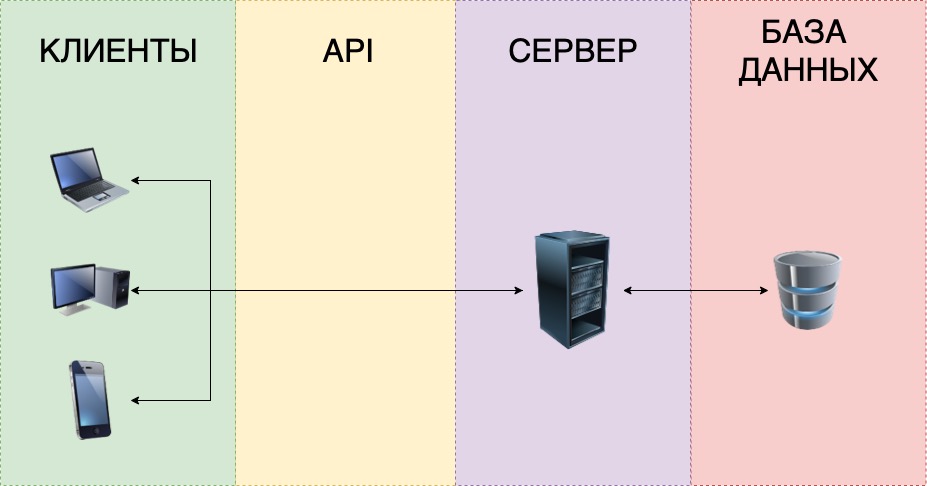


Рисунок 2.1 – клиент-серверная архитектура

В разрабатываемой программном обеспечении под клиентами понимаются специальные ПО для организации доступа в СФПЭС на основе WEB базированного доступа для пользователей в лице бюджетных и коммерческих организаций, а также для доступа администраторов. На данном слое хранится минимум логики, для отображения данных или их изменения клиент обращается к серверу.

Для универсализации организации доступа был использован формат API на основе протокола REST.

В качестве сервера используется wsgi-cервер под которым работает Django приложение, написанное на языке Python и реализующее всю бизнес-логику СФПЭС. При этом сервер является прослойкой между клиентом (отображающим данные) и базой данных (хранящей данные). Он обрабатывает запросы клиента и отвечает ему.

Для хранения, чтения и модификации данных, которые должны храниться между запросами пользователей, используется СУБД PostgreSQL изолированная от серверного кода ORM системой Django. Сервер базы данных отвечает за корректное хранение и консистентность данных, тем самым снимая данную обязанность с сервера, позволяя ему фокусироваться на бизнес-логике приложения.

## Разработка архитектуры серверной части

В рамках серверной части выделим подсистемы, отвечающие за блоки функциональности приложения:

* подсистема регистрации, авторизации и аутентификации;
* подсистема ввода данных об учреждении;
* подсистема формирования паспорта ПЭС;
* подсистема оплаты;
* подсистема подбора энергосберегающих мероприятий;
* подсистема интеллектуальной валидации ввода данных пользователей.

В зависимости от необходимости подсистемы серверной части могут взаимодействовать:

* между собой;
* с клиентской частью;
* с сервером базы данных.

На рисунке 2.2 изображена схема взаимодействия между подсистемами серверной части.



Рисунок 2.2 – Схема взаимодействия подсистем серверной части

Подсистема регистрации, авторизации и аутентификации – данный модуль отвечает за идентификацию пользователя и разграничение прав. Пользователями системы являются бюджетные организации вводящие свои данные, поэтому важно ограничивать доступ к ней и предоставлять его только владельцам данных.

Подсистема ввода данных об учреждении. Для формирования паспорта ПЭС и подбора списка энергосберегающих ресурсов необходим значительный объем данных. Для того что бы не вносить его каждый раз пользователю необходимо предоставить возможность сохранять их на сервере и получать к ним доступ, это и является обязанностью подсистемы ввода данных об учреждении. Для выполнения своей функции ей так же необходимо взаимодействовать с другими подсистемами и сервером базы данных.

Подсистема интеллектуальной валидации пользовательских данных. Поскольку, вводимые пользователем, данные могут быть некачественные или не консистентные, необходимо проверять их на достоверность и соответствие действительности. Ввиду сложности, поставленной задачи из-за разнородности данных, было принято решение интеллектуализировать данную подсистему.

Подсистема формирования паспорта ПЭС – отвечает за формирования .docx файла содержащего паспорт ПЭС, для этого подсистема берет данные из модуля хранения информации об организации.

Подсистема оплаты – отвечает за разграничение пользователей по статусу оплаты услуг и предоставления пользователям возможности оплатить услуги предоставляемые СФПЭС.

Подсистема подбора энергосберегающих мероприятий – облегчает работу пользователя за счет рекомендации энергосберегающих мероприятий при помощи выведенных эвристик. Данная подсистема используется подсистемой формирования паспорта ПЭС.

## Разработка общей структуры клиентского приложения

В рамках клиентской части выделим подсистемы:

* подсистема взаимодействия с API;
* подсистема роутинга;
* подсистема для пред показа паспорта ПЭС;
* контейнеры;
* компоненты.

В зависимости от необходимости модули клиентской части могут взаимодействовать:

* между собой;
* с пользователем;
* с серверной частью по средствам API.

На рисунке 2.3 изображена схема взаимодействия между компонентами клиентской части.

****

Рисунок 3.1 – Схема взаимодействия подсистем клиентской части

При открытии WEB-приложения в браузере пользователь использует URL ресурса, по которому осуществляется http запрос. В зависимости от адреса по которому переходит пользователь WEB-приложение должно отображать различные экранные формы и страницы, для сопоставления URL и контента, предоставляемого пользователю, используется подсистема роутинга.

При проектировании клиентской части используется компонентно-ориентированный подход, при котором пользовательский интерфейс делится на небольшие пере используемые компоненты.

Для объединения компонентов в страницы используется контейнеры, так же в контейнеры выносится вся логика управления данными для упрощения проектирования и использования компонентов.

Паспорт ПЭС представляет из себя .docx файл и для его просмотра необходимо свернуть браузер и открыть скачанный файл. Для того что бы упростить процесс и сделать его более бесшовным используется подсистема пред просмотра паспорта ПЭС, позволяющая ознакомится с содержанием сгенерированного файла внутри браузера.

Для разделения отображения данных в компонентах и манипуляций с ними используется подсистема взаимодействия с API, представляющая из себя использования паттерна адаптер и позволяющая в любой момент изменить API без изменения остальной кодовой базы.

## Формирование API приложения

Для взаимодействия между клиентской и серверными частями было принято решение использовать REST протокол, базируемый на отправке HTTP запросов.

Использование REST протокола для реализации API подразумевает предоставление клиентской части возможность отправки запросов для взаимодействия с ресурсами серверной части. В системе СФПЭС реализованы следующие типы запросов:

* GET – получение различных данных об организации, зданиях и т.д.;
* POST – создание записей (о зданиях, транспорте) и выполнение каких-либо действий (регистрация, формирование паспорта ПЭС);
* PATCH – модификация записей, созданных в результате POST запросов;
* DELETE – удаление информации из системы (сокращение количества зданий, приборов учета и т.д.).

Запросы, отправляемые ресурсам выглядят следующим образом:

* /<название ресурса> в случае, когда происходит обращение к типу ресурсов (получить все приборы учета);
* /<название ресурса>/<идентификатор ресурса> в случае, когда обращение происходит к конкретному объекту ресурсу (модификация информации об организации).

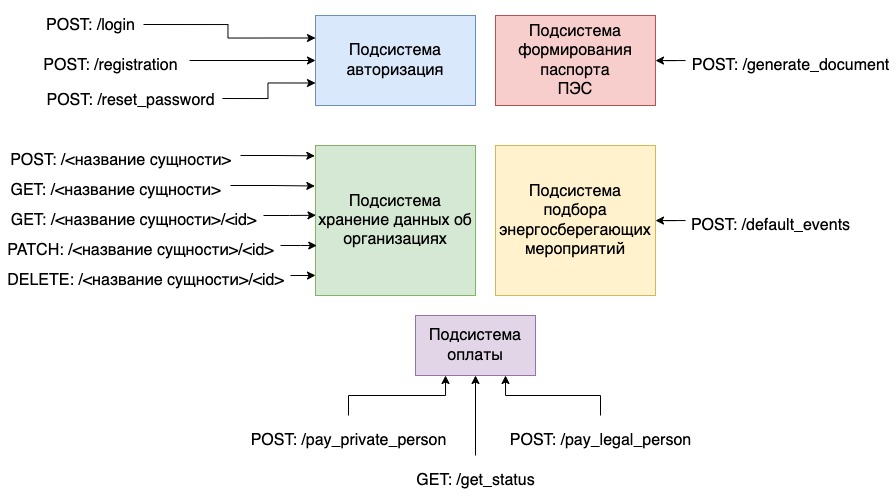


Рисунок N.N – Структура API серверной части СФПЭС.

Как видно из схемы структуры API в качестве ресурсов выступают подсистемы серверной части, которые по средствам REST API предоставляют клиентской части заложенную в СФПЭС функциональность.

**Вывод**

В данной главе были выполнены задачи разработки структуры и архитектуры СФПЭС. Выделены подсистемы и описана их функциональность.

Была разработана структура базы данных и MLP классификатор для последующей валидации данных об организации, вводимых пользователем.

Так же была сформирована структура API между серверной и клиентскими частями и выделенные основные ресурсы, используемые клиентской частью.

## РЕАЛИЗАЦИЯ СФПЭС

Подсистемы СФПЭС разбиты на модули, отвечающие за части функциональности подсистемы (рисунок N.N).

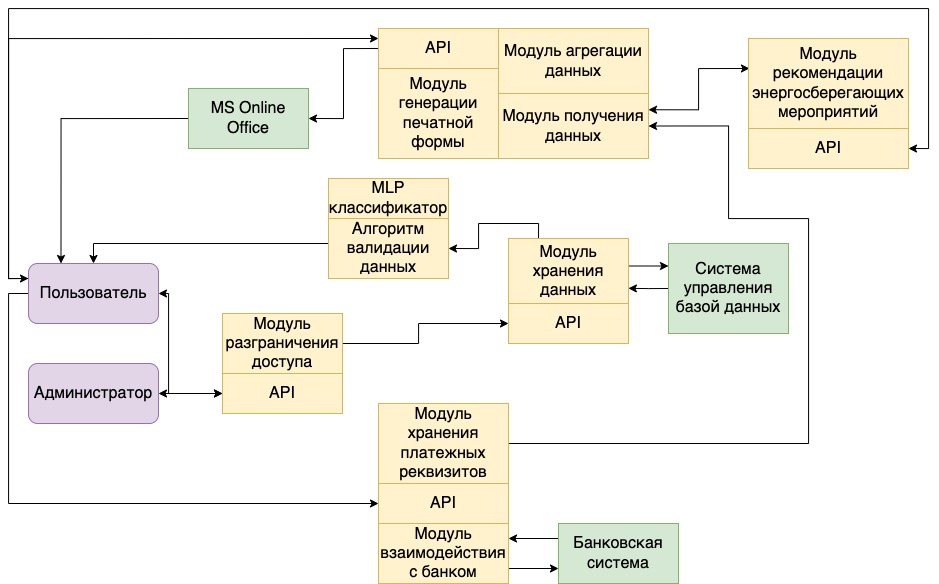


Рисунок N.N – Модульная структура СФПЭС.

Все описанные на рисунке NN модули реализованы на языке программирования Python3.10 с использованием фреймворка Django и библиотеки Django Rest Framework.

Клиентский слой системы реализован на языке программирования TypeScript с использованием библиотеки ReactJS.

## Реализация подсистемы регистрации, авторизации и аутентификации

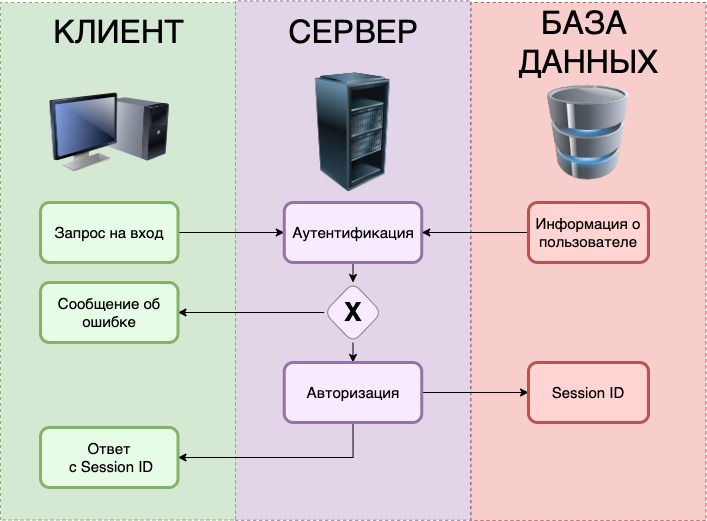


Рисунок N.N – Блок-схема авторизации и аутентификации.

Разработанная система аутентификации пользователей базируется на встроенной во фреймворк Django авторизации с помощью sessionID и базовом классе для хранения данных о пользователе User.

Для того что бы определить от чьего имени осуществляется запрос, используется идентификатор сессии, хранящийся в cookie браузера.

При входе в систему сервер генерирует идентификатор и сохраняет его в базе данных, затем отправляет его на клиент в виде ответа, а клиент получив идентификатор сохраняет его в cookie.

При отправке запроса клиент добавляет идентификатор в заголовки запроса, он в свою очередь проверяется на соответствие с хранящимся в базе данных.

При выходе из системы идентификатор, хранящийся в базе данных, удаляется.

## Реализация модуля разграничения доступа

В качестве подхода для реализации модуля разграничения доступа использована ролевая модель. С предоставлением роли для доступа к конкретной бюджетной организации (RoleUser) или для предоставления доступа к определенной группе бюджетных организаций (RoleAdmin).

Для хранения информации о пользователе, которая не предусмотрена в базовом классе User, создана промежуточная модель Employer, хранящая ссылку на класс User и содержащая дополнительные поля:

* first\_login – был ли осуществлен первый вход в систему;
* last\_change\_password – когда последний раз был изменен пароль.

class Employer(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

first\_login = models.BooleanField(default=True)

last\_change\_password = models.DateField(default=None, blank=True, null=True)

Для представления информации о каждой конкретной роли, создаются модели наследующие свойства базовой модели Role, которая ссылается на Employer. RoleUser хранит внешний ключ на конкретную организацию, существование RoleUser между пользователем и организацией предоставляет пользователю доступ к ней.

class Role(models.Model):

employer = models.ForeignKey(Employer, on\_delete=models.SET\_NULL, blank=True, null=True)

class RoleUser(Role):

eobject = models.ForeignKey(EObject, on\_delete=models.CASCADE)

Реализованная система ролей предоставляет большую гибкость в выдаче прав, т. к. позволяет как нескольким пользователем иметь доступ к одной и той же организации, так и одному пользователю иметь доступ к множеству организаций.

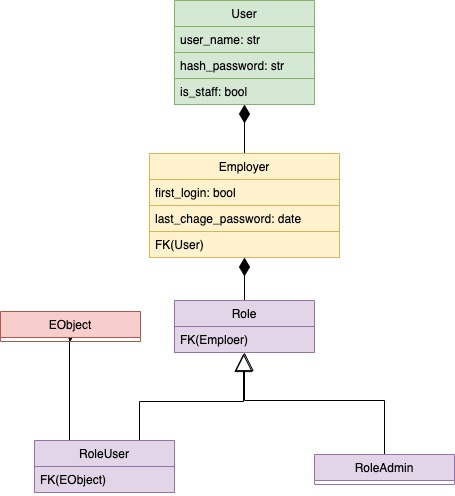


Рисунок 2.3 – Диаграмма классов системы ролей

## Реализация модуля API

Для реализаций функций авторизации и регистрации было разработано следующее API.

POST: /accounts/login

В качестве параметров принимает две строки username(логин) и password(пароль). Для авторизации и входа используются стандартные функции login и authenticate. Реализация API описана ниже

def login(self, request):

try:

user = authenticate(username=username,password=password)

except:

user = None

if user is not None:

login(request, user)

return Response(username+"authorized",status=status.HTTP\_200\_OK)

Функция authenticate принимает в качестве аргументов логин и пароль и возвращает объект User в случае успешной аутентификации или возвращает ошибку.

Функция login принимает объект запроса и объект User, после чего добавляет идентификатор сессии и отправляет ответ для его сохранения на стороне клиента.

POST: /accounts/logout

Реализация API описана ниже

def logout(self, request):

logout(request)

return Response("Logout", status=status.HTTP\_200\_OK)

Отправляется без параметров и выполняет выход из системы для пользователя с помощью встроенной функции logout.

POST: /accounts/register

Реализация API описана ниже

def register(self, request):

serializer = UserSerializerCreate(data=request.data)

user = serializer.save()

employer=Employer.objects.create(user=user)

login(request, user)

return EObjectController.create\_first\_eobject(employer)

Принимает в качестве параметров логин и электронную почту пользователя. В качестве результата создает для пользователя аккаунт и объект организации с пустыми данными. После этого отправляет пользователю на почту письмо с информацией для входа в систему.

## Реализация разделения доступа к API приложения

Разработанная система ролей используется для разграничения доступа к данным. В классе ModelViewSet, используемом для представления группы API, предусмотрено поле permission\_classes, в которое необходимо поместить список классов наследуемых от BasePermission.

При получении запроса фреймворк вызывает у них методы has\_permission и has\_object\_permission принимающий в качестве аргумента объект запроса, хранящий все необходимые данные для предоставления доступа:

- тип запроса;

- данные, содержащиеся в запросе;

- информация о пользователе, отправившем запрос.

Пример класса, разграничивающего доступ к объектам, хранящим информацию об организации:

class EObjectPartPermission(BasePermission):

def has\_object\_permission(self, request, view, obj):

controller = UserPermissionController(request.user)

return obj.eobject in controller.get\_avalible\_eobjects()

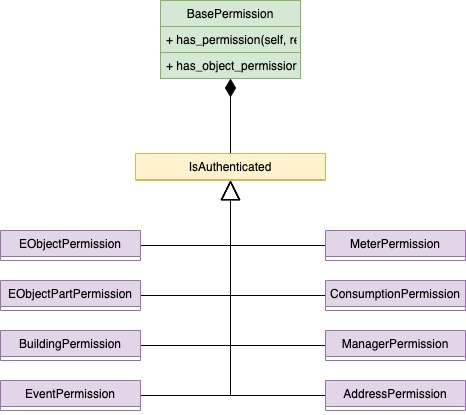


Рисунок 2.4 – Диаграмма классов, разграничивающих доступ к API подсистемы хранения данных об организации

Все классы в данной иерархии наследуют IsAuthenticated, данный класс определяет, что для доступа к подсистеме пользователь обязан осуществить вход и авторизоваться в системе.

## Реализация подсистемы хранения данных об учреждении

## Реализация модуля моделей

Для каждой сущности были разработаны соответствующие модели, описанные на языке Python для встроенное во фреймворк Django ORM. ORM позволит обращаться к сущностям базы данных и строить запросы с помощью Python, тем самым создавая простую абстракцию для работы с базой данных.

Так же это позволит отделить хранение данных от их использования и менять сервер базы данных при необходимости, без изменения кодовой базы.

Со списком моделей, их полей и описанием можно ознакомиться в приложении А.

Описание моделей на языке Python происходит в декларативном стиле. Для этого создается класс наследник Model и в качестве полей указываются столбцы базы данных. В зависимости от типа поля, его конструктор принимает различные параметры такие как текстовое описание поля или значение по умолчанию. Ниже указан пример описания модели годового потребления энергоресурса.

class Consumption(models.Model):  
 resource = models.ForeignKey(Resource, on\_delete=models. CASCADE, null=True,blank=True,related\_name='consumptions')  
 year = models.IntegerField(u"Год потребления", default = 2020)  
 annual\_rate = models.FloatField(u"Фактическое потребление за год (тыс.л)", default=0)  
 annual\_cost = models.FloatField(u"Фактическая стоимость за год (тыс.руб)", default=0)

После декларативного описания моделей Django ORM необходимо вызвать команду makemigrations для создания файлов миграций, которые в последствии при вызове команды migrate будут необходимы для создания структуры БД в соответствии с моделями, описанными ранее.

## Реализация модуля API

Для описанных моделей было реализовано CRUD API:

* create (создание);
* read (чтение);
* update (обновление);
* delete (удаление).

Для моделей описывающие различные типы (окон, утепления, материала стен и т. д.) предусмотрено только API для чтения, т. к. они используются всеми пользователями системы, их создание, редактирование и удаление осуществляет администратор при инициализации системы или в процессе ее эксплуатации.

Для оптимизации построения API использовался базовый класс фреймворка Django Rest Framework – ModelViewSet. Он позволяет в декларативном стиле описать реализуемое API с помощью следующих полей:

* queryset – список объектов доступных в рамках API;
* http\_method\_names – список http запросов, которые может принимать данный класс;
* serializer\_class – класс используемый для сериализации и десерализации объектов в формат json и обратно соответственно;
* permission\_classes – cписок классов разграничивающих доступ к API.

Список разработанных API:

* /dict/\* - GET запросы используемые для получения списков типов;
* /user\_lk/address – управление адресами;
* /user\_lk/available\_events/get\_available\_event\_for\_eobject – GET запрос для получения списка энергосберегающих мероприятий, которые можно провести в рамках объекта;
* /user\_lk/building – управление информацией о зданиях;
* /user\_lk/eobject – управление информацией об организации;
* /user\_lk/events – управление информацией об энергосберегающих мероприятиях;
* /user\_lk/manager – управление информацией о директорах (менеджерах) организации;
* /user\_lk/meter – управление информацией о приборах учета;
* /user\_lk/resource/consumption – управление информацией о потреблениях энергетических ресурсов;
* /user\_lk/vehicle – управление информацией о транспортных средствах;

## Реализация подсистемы формирования паспорта ПЭС

Подсистема формирования паспорта ПЭС используя данные из подсистемы хранения информации об организации генерирует .docx документ. Для выполнения задачи и сохранения максимальной гибкости подсистемы вся логика поделена на следующие модули:

* получение данных из подсистемы хранения информации об организации;
* агрегирование полученной информации для удобства заполнения шаблона документа;
* заполнение шаблона документа.

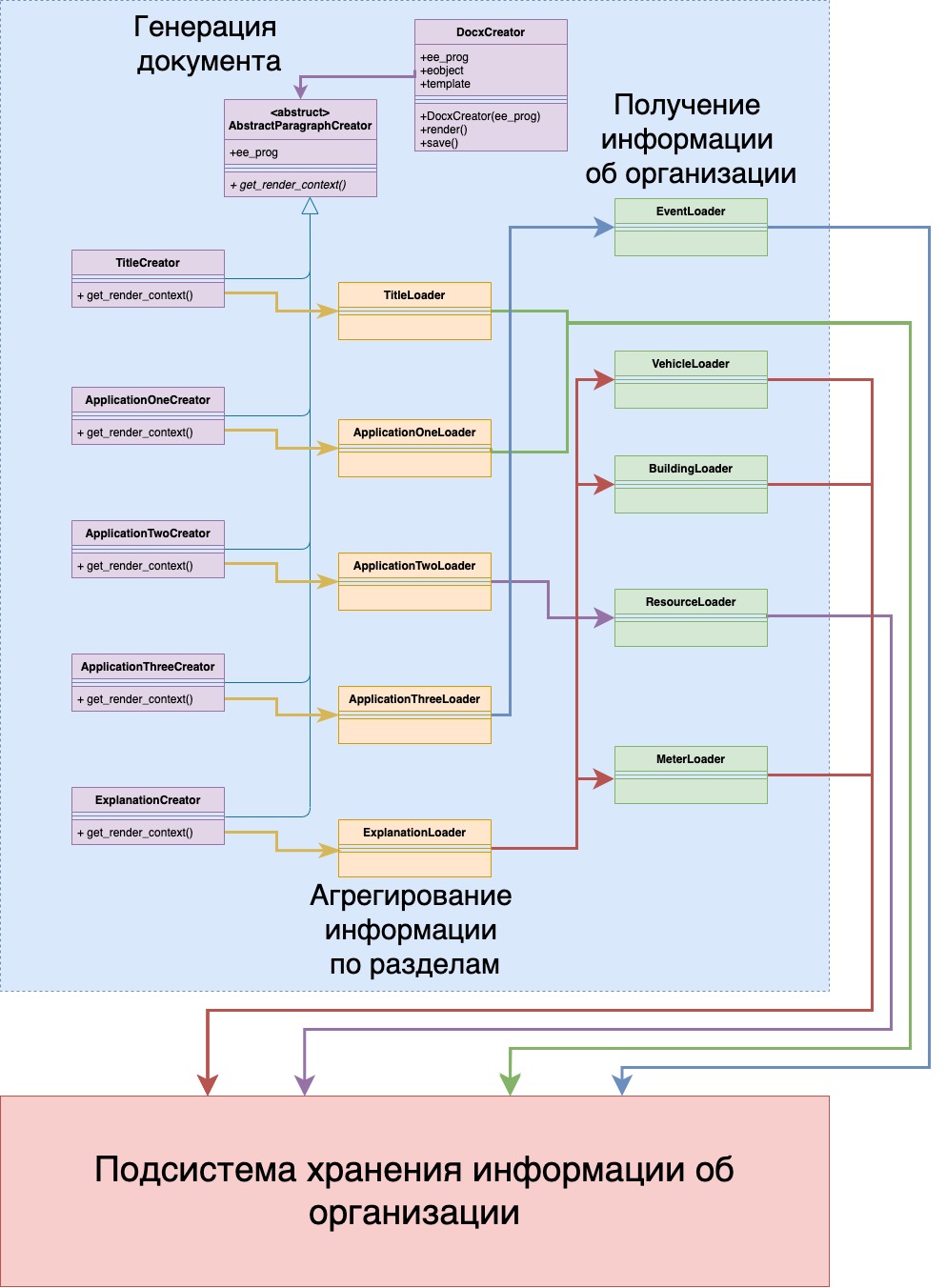


Рисунок 2.7 – Общая структура подсистемы

На схеме цветом выделены модули подсистемы:

* фиолетовый - модуль заполнения шаблона паспорта ПЭС;
* желтый – модуль агрегирования полученной информации;
* зеленый – модуль получения данных из подсистемы хранения информации об организации.

## Реализация модуля получения данных из подсистемы хранения информации об организации

Для каждой модели, хранящей информацию об организации, были разработаны промежуточные классы (Loader), в конструкторе они получают объект модели и затем предоставляют интерфейс, предоставляющий всю необходимую информацию для агрегатов.

Данный подход используется для разделения формата хранения информации и ее использования, в результате данного решения можно изменять подсистему хранения информации об организации без значительных изменений в подсистеме формирования паспорта ПЭС.

Ниже представлен пример одного из Loader класса предназначенного для получения информации о приборе учета.

class MeterLoader:

def \_\_init\_\_(self, meter: MeterPES):

self.meter = meter

@property

def place(self) -> str:

if self.meter.building is None:

return "Неизвестно"

info\_loader = BuildingLoader(self.meter.building)

return info\_loader.name

@property

def verification\_info(self) -> str:

return ", ".join([str(self.meter.year\_commissioning),

str(self.meter.year\_verification),

str(self.meter.quarter\_verification)])

Предлагаемый подход так же полезен тем, из-за того, что не вся необходимая информация хранится в базе данных в требуемом виде и для ее использования производится модификация, как в методах verification\_info, place и interval.

Все методы классов типа Loader отмечены декоратором @property позволяющем обращаться к методу как к полю класса без использования конструкции для вызова функции.

## Реализация модуля агрегирования данных для разделов

В ходе анализа предметной области было выяснено, что паспорт ПЭС состоит из пяти частей:

* заголовка;
* трех приложения;
* дополнительной и справочной информации.

Для удобства заполнения шаблона и генерации документа было принято решение выделить классы агрегирующую информацию, полученную на предыдущем слое. По своей структуре они похожи на обычные классы типа Loader, но в результате их работы информация структурирована в более крупные объекты, которые и будут использоваться при генерации документа.

Использование классов для агрегирования информации позволяет соединить большое количество разнородной информации в цельные и консистентные объекты более подходящие для использования в языке разметки шаблонизатора jinja2 используемом на следующем этапе. [ВСТАВИТЬ ССЫЛКУ НА Jinja]

## Реализация модуля заполнения шаблона паспорта ПЭС

На данном этапе необходимо внести полученную из агрегаторов информацию в .docx документ.

Для упрощения задачи используется библиотека языка Python3 – docxtpl. Данная библиотека представляет из себя обертку над шаблонизатором jinja2 и позволяет, разметив docx файл, легко заполнить его данными. [ВСТАВИТЬ ССЫЛКУ НА docxtpl]

Для заполнения шаблона данными происходит в четыре этапа:

* создание объекта типа DocxTemplate из библиотеки docxtpl (конструктор принимает путь к файлу, содержащему шаблон);
* получение данных из агрегаторов в виде объекта типа dict языка Python;
* передача объекта в метод render объекта DocxTemplate;
* сохранение файла с помощью метода save.

Перед тем как выполнять заполнение шаблона его так же необходимо создать и разметить. Формат разметки напоминает любой другой язык разметки типа html. Внутри файла имеется доступ к объекту, который был передан в функцию render.

Обращение к полям объекта аналогично обращению в языке python, за исключением того, что его необходимо обрамлять в двойные фигурные скобки.

Так же язык разметки поддерживает и другие стандартные конструкции языков программирования такие как if else, for и т. д., их для использования необходимо обрамлять в конструкцию {% %}.

## Реализация модуля API

В качестве API данная подсистема предоставляет один GET HTTP запрос принимающий на вход идентификатор организации и отправляющий в ответ файл составленного паспорта ПЭС. В случае нехватки каких-либо данных API возвращает код ошибки 400 и описание информации, которую необходимо заполнить для корректной генерации.

## Реализация подсистемы оплаты

СФПЭС предоставляет пользователю два варианта оплаты, предоставляемых услуг. Оплата в роли физического лица подразумевает взаимодействие с банковской системой. В роли юридического лица составляется договор оказания услуги, требующий платежные реквизиты заказчика. Исходя из этого были выделены два модуля и API, который предоставляет единую точку входа для оплаты (Рисунок N.N).

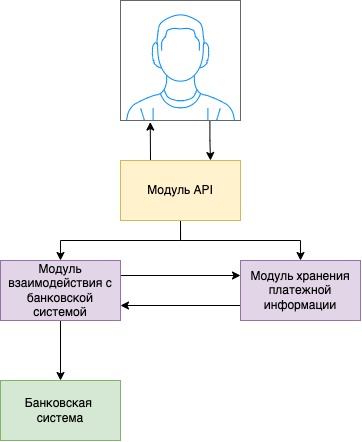


Рисунок N.N – Общая структура подсистемы

Модуль взаимодействия с банковской системой используется для оплаты физическими лицами, а также использует модуль хранения платежной информации для сохранения результата транзакций.

## Реализация модуля хранения платежных реквизитов

Для реализации модуля хранения платежных реквизитов использовался тот же подход, что и в подсистеме хранения данных об организации.

Данный модуль содержит две модель PayInfo и Pay.

PayInfo содержит платежные реквизиты клиента, такие как номер лицевого счета, БИК банка, название банка и другую информацию. Она необходима для заполнения договора на оказание услуг.

Модель Pay является представлением транзакции, созданной в банковской системе. Она содержит стоимость платежа, id транзакции и другие служебные поля из банковской системы. Так же модель Pay хранит информацию о статусе транзакции, он может быть одним из:

* создан в системе;
* создан в платежной системе;
* ошибка при создании в платежной системе;
* подтвержден;
* возврат.

В силу большого количества статусов и наличия у пользователя возможности открывать несколько транзакций единовременно, был разработан контроллер позволяющий инкапсулировать логику вычисления статуса пользователя, а именно оплатил ли он возможность скачать полную версию паспорта ПЭС.

class ControllerPayEmployer:

def \_\_init\_\_(self, emp: Employer):

self.emp = emp

def is\_allowed\_download(self):

pays = Pay.objects.filter(user=self.emp, status=3)

if pays:

return True

return False

В качестве входных параметров контроллер получает корневую модель ролевой системы Employer на которую ссылаются модели транзакций Pay. И возвращает Истину, если у пользователя emp существуют транзакции со статусом 3 (Подтвержден).

## Реализация модуля взаимодействия с банковской системой

В качестве банковской системы был выбран банк Тинькофф, обладающий простым Rest API для взаимодействия. Создание транзакций, а также получение информации о ее статусе происходит с помощью Http запросов. Для их отправки используется библиотека requests из стандартной поставки языка программирования Python. Для функционирования модуля используются два запроса: создание транзакции и получение информации о транзакции (рисунок N.N).



Рисунок N.N – Процесс оплаты в роли физического лица

Как видно из рисунка сервер выступает в роли API для использования клиентом банковской системы и сохраняет информацию о транзакциях в базе данных. Данный подход к реализации подсистемы оплаты позволяет не зависеть от используемой банковской системы и в любой момент сменить ее на более выгодную.

Функции создания транзакции и получения информации о ней реализуются с помощью контроллера PayEmployer с двумя методами refresh\_status и create\_pay.

## Реализация модуля API

В качестве API система предоставляет пользователю:

* возможность сохранять платежную информацию (CRUD /pay/pay\_info);
* инициировать оплату в виде физического лица (POST: /pay/private\_pay);
* инициировать оплату в виде юридического лица (POST: /pay/legal\_pay).

Для обработки запросов инициации оплаты использовались наследники ViewSet с методами отмеченными декораторами actions. Данные запросы не содержат параметров, т. к. для инициации запроса необходима информация о пользователе, а она содержится в объекте request передаваемом в качестве параметра метода.

@action(detail=False, methods=['post'], name='private\_pay')

def private\_pay(self, request):

user = request.user

controller\_pay = ControllerPays()

controller\_pay.create\_pay(emp=user.employer,

customer\_email=user.email)

return Response({

'url':controller\_pay.execute\_pay()

}, 200)

CRUD API для хранения платежных реквизитов (изменение модели PayInfo) разработан аналогично CRUD API подсистемы хранения данных об организации, с помощью базовых классов ModelViewSet и ModelSerializer.

## Алгоритм подбора энергосберегающих мероприятий

Для предоставления функции подбора энергосберегающих мероприятий разработана рекомендательная система, действующая по следующему алгоритму (рисунок NN).

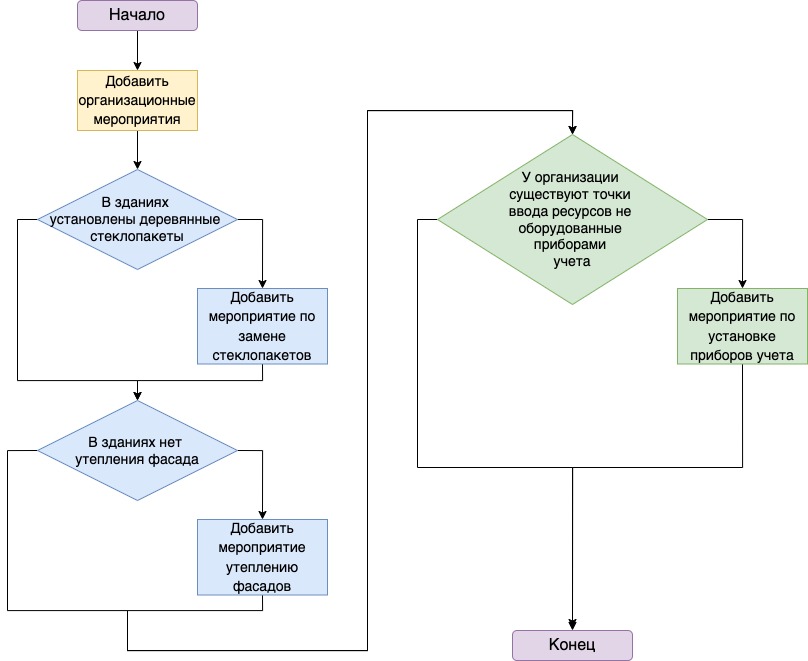


Рисунок 2.9 – Пример некоторых эвристик для рекомендации энергосберегающих мероприятий

На рисунке изображены некоторые блоки рекомендательной системы. При реализации данной подсистемы, был разработан класс EventAdvicer.

class EventAdviser:

def \_\_init\_\_(self, eobject: EObject):

self.eobject = eobject

...

def generate\_advice\_events(self) -> list:

buildings = self.eobject.buildings.filter()

list\_advices = [Event.objects.create(percentage\_saved=0,

cost\_event=0,

building=buildings.first(),

year=self.eobject.base\_pes\_year + 1,

type\_event=DictTypeOfEvents.objects.get(type=35))]

list\_advices += self.\_get\_heating\_advice\_events\_type()

list\_advices += self.\_get\_input\_advice\_events\_type()

return list\_advices

При создании EventAdvicer принимает объект из подсистемы хранения данных об организации и на основе него делает предположения о том, какие энергосберегающие мероприятия будут наиболее эффективны. Затем, при вызове метода generate\_advice\_events, формирует список рекомендуемых энергосберегающих мероприятий.

В силу того, что системе не известна стоимость внедрения энергосберегающего мероприятия (она зависит от множества внешних факторов), после создания рекомендуемых мероприятий пользователь заполняет эту информацию в ручную при помощи подсистемы хранения данных об организации.

## Реализация подсистемы валидации данных об организации на основе MLP классификатора

## Реализация MLP классификатора

В качестве реализации ИНС была создана модель многослойного персептрона. Код модели на языке Python с использованием библиотеки Keras [15, 16] ВСТАВИТЬ ССЫЛКУ НА КЕРАС представлен ниже:

model = Sequential()

model.add(Dense(56, input\_dim=28, activation='elu'))

model.add(Dense(112, input\_dim=56, activation='elu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', metrics=['accuracy'])

В качестве модели используется последовательная модель (Sequential). В качестве слоев плотные слои (Dense), каждый выходной нейрон которых связан с входным нейроном следующего слоя. Так же Keras позволяет указывать для каждого слоя активационную функцию (параметр activation).

После формирования структуры MLP в декларативном стиле, ИНС собирается с помощью метода compile. В него передаются дополнительные параметры для сборки, в том числе список метрик для оценки работы ИНС.

## Алгоритм валидации данных об организации на основе MLP классификатора

Разработанный MLP классификатор применяется для валидации данных и встроен в процесс ввода данных пользователем.



Рисунок 2.9 – Алгоритм валидации данных об организации

После ввода данных об организации информация поступает в подсистему валидации данных. В результате анализа MLP классификатор определяет корректность и консистентность данных.

Используя результат анализа подсистема хранения данных решает сохраняет информацию и уведомляет пользователя в случае наличия ошибок во введенной информации.

## Результаты работы подсистемы

На рисунке 2.11 изображены результаты обучения модели на выборке из 1912 верифицированных образов.

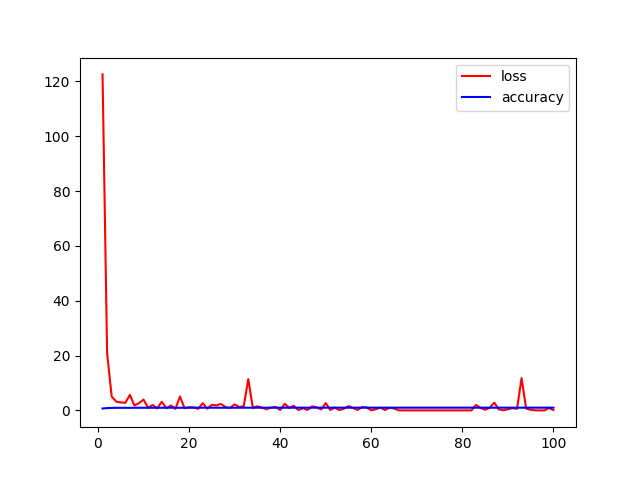


Рисунок 2.11 – Модель многослойного персептрона, выполняющего бинарную классификацию

Обучение проходило в 100 эпох и размером пакета данных, равным 239. Также на рисунке отражаются такие характеристики ИНС, как точность (accuracy) и потери (loss) для каждой эпохи.

В качестве тестовых данных было взято 4000 образов, хранящих информацию о бюджетных организациях.

Процент ложноположительного срабатывания (0.28%) и ложноотрицательного (0.5%) относительно мал. Точность классификации равна 0.997.

Приведенные результаты точности распознавания (99.7%) разработанным классификатором на основе многослойного персептрона говорят о том, что данный подход может нести только рекомендательный характер, следовательно пользователь в любом случае должен перепроверять вводимые данные, но разработанная подсистема поможет обнаруживать ошибки и оперативно их исправлять.

## Реализация клиента СФПЭС

Клиент СФПЭС представляет из себя Web-базированный интерфейс с доступом через браузер, реализованный с помощью языка программирования TypeScript и библиотеки ReactJS. Для отправки запросов используется встроенная в браузер функция fetch.

## Роутинг

Для доступа к экранным формам используется URL ресурса в сети интернет, получаемого с помощью http запроса.

В зависимости от URL пользователь видит различные экранные формы, так же некоторые страницы могут быть недоступны для определенной группы пользователей. Решением данной задачи занимается подсистема роутинга.

Для разграничения прав реализован класс Routes. При инициализации он принимает на вход объект пользователя, полученный через API и хранящий информацию о текущей роли пользователя. В зависимости от роли пользователя формируется список URL доступных ему.

const { currentRole } = user;

const routes = [

{

title: 'Главная',

href: '/',

},

];

if (currentRole === C.ROLES.ADMIN) {

routes.push(...[

{

title: 'Управление организациями',

href: '/admin\_lk/eobjects',

},

]);

}

return routes;

Для сопоставления URL и экранной формы используется компонент RouteList. При инициализации каждому URL из Routes устанавливается соответствующий контейнер, который будет отображаться пользователю при переходе по нему.

return routes.map(({ path, Component }) => (

<Route

path={path}

element={

allowedHrefs.includes(path)

? <Component />

: <NoAccess />

}

key={path}

/>

));

В случае, когда пользователь переходит по недоступному или несуществующему URL, клиент отображает компонент NoAccess, в котором говорится о том, что ресурс недоступен или не найден.

## Компоненты и контейнеры

Экранные формы клиента декомпозированы на компоненты и контейнеры по принципу разделение ответственности за хранение и использование данных.

Контейнеры хранят данные, полученные по средствам API и передают часть информации компонентам (Рисунок NN).



Рисунок 2.9 – Структура контейнеров и компонентов форм ввода данных об организации и зданиях

Данный подход позволяет пере использовать разработанные компоненты и использовать некоторые контейнеры в качестве компонентов, тем самым значительно упрощая разработку и сокращая объем кодовой базы.

## Взаимодействие с API

Для взаимодействия с API, предоставляемым сервером, клиент использует функцию fetch встроенную в браузеры.

В базовом представлении она принимает на вход URL запроса, заголовки, метод и тело запроса. Запросы, посылаемые клиентом, содержат один и тот же заголовок, а также формат представления данных – JSON. Поэтому была разработана функция обертка fetchApiSendJson.

export const fetchApiSendJson = (url, method, csrftoken, params) => fetchApi(url, {

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

'X-CSRFToken': csrftoken,

},

method: method,

body: JSON.stringify(params),

});

Помимо отправки запроса, необходимо корректно обработать ответ и вызвать исключение в случае, когда запрос не увенчался успехом. Для выполнения этой функции используется checkStatus и функция handleError.

const checkStatus = (response, type) => {

const {status} = response;

if (status >= 200 && status < 300) {

return getContentByType(response, type);

} else {

return handleError(response);

}

};

В случае получения статуса от 200 до 300 (успешное выполнение запроса) функция возвращает данные в соответствии с их типом, в ином случае генерируется исключение с описанием серверной ошибки.

В виде ответа клиент может получить следующие типы данных:

* пустой ответ;
* ответ в формате JSON;
* ответ в виде текстовой строки;
* ответ в виде Blob (двоичный объект).

**Вывод**

В данной главе были выполнены задачи реализации СФПЭС. Выделены и реализованы модули разработанных подсистем.

Был разработан клиент СФПЭС с Web-базированным доступом и налажено взаимодействие между клиентской и серверной частями.

Так же был применен MLP классификатор в реализации алгоритма валидации входных данных.

## АПРОБАЦИЯ И ТЕСТИРОВАНИЕ СФПЭС

## Система автоматического тестирования

Для повышения качества СФПЭС было автоматизировано тестирование модулей системы, а также ее API.

Тестирование СФПЭС осуществлялось с помощью библиотек pytest (реализация автотестов и их запуск) и coverage (расчет процента покрытия).

Реализованы следующие виды тестов:

* модульные;
* интеграционные.

## Модульное тестирование

Модульное тестирование используется для контроля качества и верности реализации обособленных участков кода (модулей), таких как алгоритм подбора энергосберегающих мероприятий, генерация печатной формы паспорта ПЭС и другие.

При использовании pytest, для реализации теста необходимо объявить функцию с названием содержащим test\_. При запуске автотестов pytest обнаружит все функции имеющий такой формат и выполнит их.

@pytest.mark.django\_db

def test\_create\_with\_fake\_duration(self,

controller\_ee\_program\_without\_pes,

fake\_durations\_pes):

controller = controller\_ee\_program\_without\_pes

for fake\_duration in fake\_durations\_pes:

try:

controller.create\_programm(fake\_duration)

except EEProgramDurationException as e:

continue

assert False, "Была создана программа с неверной продолжительностью."

Для проведения тестирования необходимо использовать набор тестовых данных, описываемый с помощью декоратора pytest.fixtures().

@pytest.mark.django\_db

@pytest.fixture

def controller\_ee\_program\_without\_pes():

controller = ControllerEEProgram(

EObject.objects.get(pk=8080),

start\_year=2021

)

controller.delete\_programm()

return controller

Для реализации СФПЭС был использован подход разработки через тестирование, что существенно увеличило качество тестирования продукта (таблица NN).

Таблица N.N – Параметры набора данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Название подсистемы** | **Процент покрытия тестами, %** |
| 1 | подсистема регистрации, авторизации и аутентификации | 93% |
| 2 | подсистема ввода данных об организации | 91% |
| 3 | подсистема валидации данных об организации | 89% |
| 4 | подсистема формирования паспорта ПЭС | 94% |
| 5 | подсистема оплаты | 90% |
| 6 | подсистема подбора энергосберегающих мероприятий | 91% |

Для оценки покрытия подсистем модульными тестами использовалась библиотека coverage (pytest-cov). Как видно из таблицы были получены высокие показатели покрытия модульными тестами от 89% до 94%.

## Интеграционное тестирование

В качестве интеграционного тестирования проведено тестирование API СФПЭС.

Для тестирования API использован класс предоставляемый фреймворком Django в пакете test, а именно Client. Он позволяет пройти аутентификацию по логину, без использования пароля – force\_login, и затем отправлять http запросы от имени пользователя.

self.client = Client()

self.client.force\_login(

User.objects.get(username="Test12345678")

)

Класс Client позволяет эмулировать работу клиента СФПЭС и проводить интеграционное тестирование модулей, используемых при реализации API.

def test\_without\_title(self):

response\_without\_title = self.client.post(

"/api/user\_lk/feedback/send\_feedback",

{

"description": self.\_test\_description\_feedback

}

)

assertEqual(

response\_without\_title.status\_code,

status.HTTP\_400\_BAD\_REQUEST

)

assertEqual(

response\_without\_title.data["detail"],

"Укажите Заголовок"

)

Интеграционные тесты применяются к Django приложению rest\_ee, которое реализует API CФПЭС. Покрытие тестами так же было оценено с помощью утилиты coverage.

В ходе анализа была выявлено что покрытие кодовой базы интеграционными тестами составляет ~91%, что является достаточным для СФПЭС.

## Представление графа интерфейса

В ходе разработки и реализации клиента СФПЭС были выделены и реализованы следующие экранные формы (рисунок NN):

* страница авторизации;
* страница регистрации;
* страница смены пароля;
* страница ввода данных об организации;
* модальное окно предпросмотра ПЭС;
* модальное окно обратной связи;
* модальное окно оплаты.

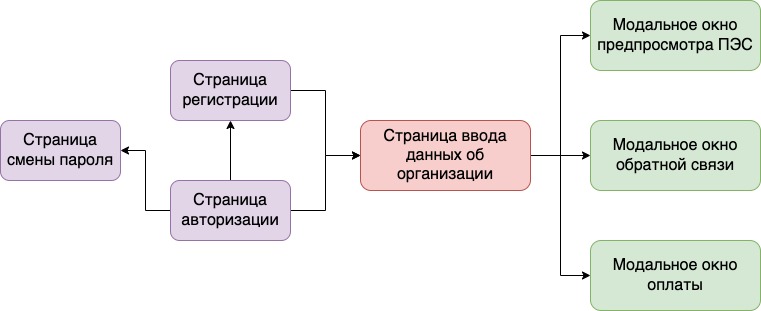


Рисунок 2.9 – Граф переходов между экранными формами клиента СФПЭС

Из рисунка видно, что все формы делятся на три группы:

* страницы связанные с авторизацией и аутентификацией;
* страница для ввода данных об организации (Рисунок NN);
* модальные окна (Рисунок NN).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.9 – Страница ввода информации об организации

Страница ввода информации об организации представляет из себя набор сворачивающихся форм, отвечающих за различные блоки данных, хранящихся в подсистеме хранения информации об организации.

На всех страницах системы в правом нижнем углу расположены кнопки реализующие необходимые пользователю функции, такие как предпросмотр паспорта ПЭС, ввод платежных реквизитов и отправка сообщения об ошибке.

В левом верхнем углу пользователю сообщается о статусе оплаты услуг сервиса.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.9 – Модальное окно предпросмотра паспорта ПЭС

Использование модальных окон для реализации различных функций позволяет исключить переходы между страницами приложения и сделать процесс пользования сервисов бесшовным и удобным.

## Анализ технических характеристик работы ПО

## Технические характеристики

В ходе апробации СФПЭС было выяснено, что для бесперебойной работы системы необходимы следующие минимальные требования.

Требование к серверному аппаратному обеспечению:

* архитектура 64-битный ПК (amd64);
* не менее 4 физических ядер, не менее 2 Ггц;
* 120 GB Места на жестком диске;
* 8 GB RAM;
* канал предоставление доступа в Интернет 10 MБит/с.

Требования к программному обеспечению (сетевому окружению серверной аппаратной части).

Операционная система: Ubuntu 20.04.

Требование к окружению:

* python 3.6.4;
* nginx 1.16;
* Django 3.0.6;
* djangorestframework 3.11.0;
* docxtpl 0.10.0;
* gunicorn.

Требование к аппаратному обеспечению АРМ клиента, не хуже:

* тактовая частота процессора – не менее 2,4 ГГц;
* количество ядер процессора – не менее 2;
* объем оперативной памяти – не менее 4 Гб;
* объем жесткого диска – не менее 500 Гб;
* поддержка Ethernet;
* диагональ монитора – не менее 21 дюймов.

Требование к ПО пользователя: браузер с поддержкой ES6.

## Статистика работы системы за 3 месяца

При анализе рынка был проведен опрос среди восьмидесяти лиц ответственных за энергосбережение в бюджетных организациях (Рисунок NN).



Рисунок 2.9 – Результаты опроса лиц ответственных за энергосбережение

По результатам опроса видно, что 72% респондентов согласны использовать платную версию СФПЭС, из них 5% согласны использовать сервис при любой рыночной цене из-за удобства и простоты предлагаемого решения.

Так же респондентам было предложено оставить отзывы и предложения по работе СФПЭС. Из восьмидесяти респондентов трудности при работе с сервисом возникли у двоих, но благодаря оперативной работе поддержки они были решены. Остальные лица ответственные за энергосбережение отметили удобство, простоту и скорость предлагаемого решения.

В ходе работы СФПЭС как Saas сервиса за 3 месяца 670 организаций зарегистрировали аккаунты. Из них 113 пользователей произвели оплату услуг: 21 организация с помощью юридического лица и 92 в виде физического лица. Около 90% пользователей заполнили основные данные об организации необходимые для формирования паспорта ПЭС.

**Вывод**

В данной главе были выполнены задачи апробации и тестирование СФПЭС. Проведено модульное и интеграционное тестирование, покрывающее ~90% кодовой базы приложения.

Так же построен граф переходов экранных форм клиентской части приложения, протестирован и проанализирован реализованный пользовательский интерфейс.

В ходе апробации и тестирования были установлены минимальные технические требования для обеспечения бесперебойной работы серверной и клиентской частей, описано необходимое системное окружение для работы СФПЭС.

Была проведена апробация СФПЭС в рамках сервиса, распространяемого по модели Saas, указаны результаты опроса целевой аудитории о платежеспособности. Так же подведены итоги работы сервиса за три месяца.

# ЗAКЛЮЧЕНИЕ

# СПИСOК ЛИТЕРAТУРЫ

1. Экономическая теория / ред. Видяпин В.И. Добрынин А.И., Журавлёва Г.П., Тарасевич Л.С. – Москва : ИНФРА-М, 2019. – 714 с. – ISBN 978-5-16-003507-9
2. Бурмистров А.Н. Формирование модернизационной стратегии развития предприятий пищевой промышленности / авт. Бурмистров А.Н. – Саратов, 2011. – 162 с.
3. ГОСТ Р 53041-2008. Изделия кондитерские и полуфабрикаты кондитерского производства [Текст]. – Москва: Стандартинформ, 2019. – 16 с.
4. Кондитерская промышленность. [Электронный ресурс]. Кондитерская промышленность на территории России. – Электрон, дан., 2019. – Режим доступа: <https://fabricators.ru/article/konditerskaya-promyshlennost>
5. Бутейкис Н. Г. Технология приготовления мучных кондитерских изделий / Н. Г. Бутейкис, А.А. Жукова – 3-е изд., стереотипное – Москва: ACADEMA, 2015. – 302 с. – ISBN 978-5-7695-6733-9
6. Толкаченко, А. А. Правовая (судебная) бухгалтерия: курс лекций / А. А. Толкаченко, К. В. Харабет; ред. Н. А. Петухов. – 3-е изд. – Москва: Омега-Л, 2004. – 216 с. – (Библиотека высшей школы). – ISBN 5-98119-281-0
7. Оперативный учёт. [Электронный ресурс]. Библиотека банковских документов. – Электрон, дан., 2020. – Режим доступа: http://www.bbdoc.ru
8. Хозяйственный учёт. [Электронный ресурс] Хозяйственный учёт, его сущность и значение. – Электрон, дан., 2017. – Режим доступа: https://buhbook.net/buhgalterskij-uchet/teoriya-buhgalterskogo-ucheta/hozyaystvennyiy-uchet-ego-sushhnost-i-znachenie/
9. Оперативный учёт: цели, особенности, измерители. [Электронный ресурс] Оперативный учёт. – Электрон, дан., 2017. – Режим доступа: <https://www.audit-it.ru/terms/accounting/operativnyy_uchet.html>
10. Оперативный учёт. [Электронный ресурс]. Оперативный учёт. Оперативный счёт. – Электрон, дан., 2020. ­ – Режим доступа: <https://bankfs.ru/deposits/operativnyi-schet-operativnyi-uch-t-buhgalterskii-uchet-v.html>
11. Кузнецова Л. С Технология приготовления мучных кондитерских изделий: учебн. для студ. учреждений сред. проф. образования. / Л.С. Кузнецова, М. Ю Сиданова – Москва: ACADEMIA, 2013. – 320 с. – ISBN 978-5-7695-9525-7
12. ERP-система. [Электронный ресурс]. Что такое ERP-система. – Электрон, дан., 2017. ­ – Режим доступа: <https://trinion.org/articles/chto-takoe-erp-sistema>
13. Корякин А. О. Разработка клиентской части программного обеспечения учёта производства кондитерских изделий [Отчет]. – Белгород, 2020.
14. Фаулер М. Архитектура коропративных программных приложений / М. Фаулер – Москва : Издательский дом «Вильяме», 2018. – 540 с. – ISBN 978-5-6041394-0-0
15. Язык программирования Java SE 8. Подробное описание / Д. Гослинг, Б. Джой, Г. Стил и др. – 5-е изд. – Москва: «Вильямс», 2015. – 672 с. – ISBN 978-5-8459-1875-8
16. Маншин Т. С. Web-сервисы Java / Т. С. Маншин – Санкт-Петербург: БХВ-Петербург, 2012. – 560 с. – ISBN 978-5-9775-0778-3
17. Webber J. REST in Practice / J. Webber, S. Parastatidis S., I. Robinson – O'Reilly Media, Inc., 2010. – First Edition – 448 c. – ISBN: 978-0-596-80582-1
18. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг / ред. Н. С. Тригуб – 3-е изд., – Москва: Вильямс, 2017. – 1440 с. – ISBN 978-5-8459-2020-1
19. Дейт К. Введение в системы баз данных / К. Дейт / ред. К.А. Птицына / перев. К.А. Птицына. – Москва: Диалектика, 2019. – 1328 c. – ISBN 978-5-8459-0788-2
20. PostgreSQL. [Электронный ресурс]. Manual Archive. – Электрон, дан. 2020. ­ – Режим доступа: <https://www.postgresql.org/docs/manuals/archive/>
21. Национальная библиотека им. Н. Э. Баумана. [Электронный ресурс]. PostgreSQL. – Электрон, дан., 2019.  – Режим доступа: <https://ru.bmstu.wiki/PostgreSQL>
22. Уоллс К. Spring в действии / К. Уоллс / ред. Д.А. Мовчан. – Москва: ДМК Пресс, 2015. – 752 c. – ISBN 978-5-97060-171-6
23. Consul.io. [Электронный ресурс]. Habr: Consul.io, часть 1. — Электрон, дан., 2016. — Режим доступа:: <https://habr.com/en/post/278085>
24. DTO. [Электронный ресурс]. DTO vs POCO и Value Object . – Электрон, дан., 2016.  – Режим доступа: <https://habr.com/ru/post/268371/>
25. MapStruct. [Электронный ресурс]. MapStruct: Frequently Asked Questions – Электрон, дан., 2020.  – Режим доступа: <https://mapstruct.org/>
26. The Postman API Platform. [Электронный ресурс]. Create API Documentation with Postman. – Электрон, дан., 2020.  – Режим доступа: <https://postman.com/>

# ПРИЛOЖЕНИЕ А

EObject:

* ID: целочисленный, первичный ключ;
* type\_org: целочисленный, внешний ключ, ID типа организации;
* place\_adress: целочисленный, внешний ключ, ID адреса;
* director: целочисленный, внешний ключ, ID менеджера директора;
* inn: строка, ИНН;
* ogrn: строка, ОГРН;
* okved: строка, ОКВЭД;
* full\_name: строка, полное название организации;
* phone\_number: строка, номер телефона;
* pes\_having: булевой тип, наличие утвержденной программы энергосбережения и повышения энергетической эффективности.

Adress:

* ID: целочисленный, первичный ключ;
* index: строка, почтовый индекс;
* region: строка, регион;
* rajon: строка, район;
* town: строка, населенный пункт;
* building: строка, дом;
* street: строка, улица;
* corpus: строка, корпус.

Manager:

* ID: целочисленный, первичный ключ;
* last\_name: строка, фамилия;
* first\_name: строка, имя;
* position: строка, должность;
* phone: строка, номер телефона;
* email: строка, электронная почта.

Vehicle:

* ID: целочисленный, первичный ключ;
* eobject: целочисленный, внешний ключ, ID организации;
* name: строка, название;
* type\_motor\_fuel: целочисленный, внешний ключ, ID типа топлива;
* type\_brand: целочисленный, внешний ключ, ID производителя;
* count: целочисленный, количество ТС;
* carrying: число с плавающей запятой, объем грузоперевозок (тыс. т-км или пасс-км);
* milleage: число с плавающей запятой, пробег (тыс.км);
* fuel\_rate: число с плавающей запятой, количество потраченного топлива (тыс.л).

Event:

* ID: целочисленный, первичный ключ;
* eobject: целочисленный, внешний ключ, ID организации;
* type\_event: целочисленный, внешний ключ, ID типа мероприятия;
* percentage\_save: число с плавающей запятой, экономия от мероприятия (%);
* year: целочисленный, год проведения;
* cost: целочисленный, фактическая стоимость за год (тыс.руб).

TypeOfEvent:

* ID: целочисленный, первичный ключ;
* type\_resource: целочисленный, внешний ключ, ID вида энергоресурса. Может быть null если мероприятие организационного характера;
* description: строка, описание;
* default\_saved: число с плавающей запятой, базовая экономия.

Building:

* ID: целочисленный, первичный ключ;
* eobject: целочисленный, внешний ключ, ID организации;
* type\_wall: целочисленный, внешний ключ, ID типа стен;
* type\_solid\_fuel: целочисленный, внешний ключ, ID типа жидкого топлива;
* type\_wonership: целочисленный, внешний ключ, ID типа владения;
* type\_heat\_connect: целочисленный, внешний ключ, ID типа подключения отопления;
* type\_plastic\_window: целочисленный, внешний ключ, ID типа пластиковых окон;
* type\_wood\_window: целочисленный, внешний ключ, ID типа деревянных окон;
* type\_cellar: целочисленный, внешний ключ, ID типа чердака;
* type\_heat\_regulation: целочисленный, внешний ключ, ID типа управления отоплением;
* type\_ext\_wall: целочисленный, внешний ключ, ID типа материала стен;
* type\_roof: целочисленный, внешний ключ, ID типа чердака;
* type\_heat\_roof: целочисленный, внешний ключ, ID типа отопления чердака;
* name: строка, наименование здания;
* description: строка, описание;
* place: строка, местонахождение;
* square: число с плавающей запятой, общая площадь;
* space: число с плавающей запятой, общий объем;
* level\_count: целочисленный, количество этажей.

Meter:

* ID: целочисленный, первичный ключ;
* building: целочисленный, внешний ключ, ID здания;
* resource: целочисленный, внешний ключ, ID энергетического ресурса;
* meter\_model: строка, тип и марка прибора;
* accuracy\_class: строка, класс точности";
* date\_commissioning: дата, дата ввода в эксплуатацию;
* factory\_number: строка, заводской номер;
* date\_verification: дата, дата последней поверки.

Resource:

* ID: целочисленный, первичный ключ;
* eobject: целочисленный, внешний ключ, ID организации;
* type\_resource: целочисленный, внешний ключ, ID вида энергоресурса;
* name\_rso: строка, наименование ресурс снабжающей организации;
* contract\_number: строка, номер контракта;
* contract\_date: дата, дата подписания контракта.

Consumption:

* ID: целочисленный, первичный ключ;
* resource: целочисленный, внешний ключ, ID энергоресурса;
* year: целочисленный, год потребления;
* annual\_rate: число с плавающей точкой, фактическое потребление за год (тыс.л);
* annual\_cost: число с плавающей точкой, фактическая стоимость за год (тыс.руб).

DictOrganizationType:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа организации.

DictTypeOfBudgetLevel:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа уровня бюджета.

DictTypeOfOwnership:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа владения.

DictTypeOfHeatSystemConnect:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа подключения системы отопления.

DictTypeOfSolidFuel:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа используемого жидкого топлива.

DictTypeOfMotorFuel:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа использования моторного топлива.

DictTypeOfHeatLoadRegulation:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа регуляции отопления.

DictTypeOfExtWallMaterial:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа материала стен.

DictTypeOfWoodWindowBlock:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа деревянных окон.

DictTypeOfPlasticWindowBlock:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа пластиковых окон.

DictTypeOfRoof:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа чердака.

DictTypeOfHeatRoof:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа отопления чердака.

DictTypeOfCellar:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа подвала.

DictTypeOfVehicleBrand:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа производителя транспорта.

DictTypeOfResource:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа энергоресурсов.

DictTypeOfEvents:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа мероприятий.

DictTypeHWAccountingType:

* ID: целочисленный, первичный ключ;
* description: строка, описание типа учета ГВС.