МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

РАЗРАБОТКА И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ “ТЕАТР”

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Сережкиной Софии Александровны

Научный руководитель
старший преподаватель _____ Н. В. Сергеева

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2024

ВВЕДЕНИЕ

Информационная система (ИС) - совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели. Она предназначена для обеспечения людей необходимой информацией, то есть для удовлетворения информационных потребностей в рамках конкретной предметной области. Результатом работы информационных систем является информационная продукция — документы, информационные массивы, базы данных и информационные услуги.

Существует большое количество областей применения ИС - начиная с банковского дела, заканчивая сферой здравоохранения. Для того чтобы корректно разработать информационную систему, необходимо построить и проанализировать бизнес-процессы, протекающие на предприятии. Если в ходе анализа окажется, что существуют бизнес-процессы, затраты на автоматизацию которых будут сильно окупаемыми, то такой бизнес-процесс необходимо автоматизировать. Целью настоящей работы является разработка информационной системы “Театр”, а именно создание веб-сайта, в котором будут реализованы авторизация и регистрация пользователей, добавление билетов в корзину. Также целью работы является моделирование бизнес-процессов театра, в результате которого будут представлены диаграммы “As Is” и “To Be” для автоматизируемых процессов, а также построены UML диаграммы, отражающие варианты взаимодействия пользователя с системой.

Для достижения поставленных целей ставятся следующие задачи:

— определить понятия методологии SADT и принципы построения IDEF0 моделей;
— рассмотреть виды диаграммы UML и способы их построения;
— проанализировать автоматизируемые бизнес-процессы;
— определить способы взаимодействия пользователя с системой;
— изучить фреймворк Django, описать структуру проекта сайта;
— описать приложения проекта;
— представить описания логики сайта, значимых модулей и классов.

Данная тема является актуальной, потому что в современном обществе любому крупному учреждению необходима информационная система, кото-

рая не только оптимизировала бы деятельность сотрудников, но и облегчала работу пользователей. В век развития технологий предприятия без информационной системы могут проиграть конкурентам, потому что, если брать в пример предметную область театра, пользователи в 90% случаях покупают билеты через сайт театра, а не вживую в кассах.

Работа состоит из введения, трех разделов, заключения и списка использованных источников, содержащего 20 наименований.

Первый раздел дипломной работы посвящен методологиям моделирования, второй - построению UML диаграмм и IDEF0 моделей. В третьем разделе освещен фреймворк “Django”, на котором написан сайт, а также представлена реализация сайта театра.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** определяются цели и задачи работы, раскрывается актуальность темы и описывается содержание.

В **первом** разделе приводится теоретическая часть работы, посвященная методологии моделирования SADT, описанию ее инструментов, а также описание языка UML.

SADT (Structured analysis and design technique) - совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. С точки зрения SADT, модель может быть сосредоточена либо на функциях системы, либо на ее объектах. SADT-модели, ориентированные на функции, принято называть функциональными моделями, представляющие с требуемой степенью детализации систему функций, которые в свою очередь отражают свои взаимоотношения через объекты системы.

В данном разделе описывается синтаксис и логика диаграмм SADT: как устроены функциональные блоки, способ передачи информации между блоками, что представляют собой дуги.

Диаграммы - главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Функциональные блоки на диаграммах изображаются прямоугольниками. Блок представляет функцию

или активную часть системы, поэтому названиями блоков служат глаголы или глагольные обороты.

На одном уровне диаграммы располагается от трех до шести блоков. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними. Управляющая информация входит в блок сверху, в то время как информация, которую обрабатывает блок, показывается с левой стороны блока и называется входом, а результаты выхода – с правой стороны. Механизм, которым может являться как человек, так и автоматизированная система, осуществляет операцию, представляется дугой, входящей в блок снизу.

Данное обозначение отражает определенные системные принципы: входы преобразуются в выходы, управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, кто выполняет функции.

Блоки SADT всегда занимают определенное место на диаграмме. Они размещаются по степени важности - по тому влиянию, которое один блок оказывает на другие блоки диаграммы. Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наименее доминирующий - в правом нижнем углу. В результате получается “ступенчатая” схема. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.

UML (Unified Modeling Language) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. Он представляет собой набор соглашений, которые предназначены для облегчения процесса моделирования и обмена информацией в проектной группе.

Цели дизайна UML:

— сократить время на усвоение информации;
— упростить общение и взаимодействие;
— обеспечить механизмы расширяемости и специализации для расширения основных понятий;
— облегчить документирование;
— обеспечить формальную основу для понимания языка моделирования.

Язык UML предоставляет стандартный способ написания проектной документации на системы, включая концептуальные аспекты, такие как бизнес-процессы и функции системы, а также конкретные аспекты, такие как выражения языков программирования, схемы баз данных и повторно используемые компоненты ПО. UML имеет множество способов использования. К ним относятся рисование диаграмм, обмен информацией, спецификация систем, генерация кода, имитационное моделирование.

В работе описываются инструменты для построения диаграмм. Представлены диаграммы вариантов использования (Use Case Diagram), последовательности (Sequence Diagram) и активности (Activity Diagram). Благодаря диаграммам UML можно графически представлять и моделировать информационные системы, четко определяя действия системы, роли, различные взаимосвязи. Каждый вид диаграмм UML предназначен для конкретных задач и разных целей моделирования.

Диаграмма прецедентов или диаграмма вариантов использования (use case diagram) в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Она необходима для моделирования бизнес-процессов организации и требований к создаваемым системам.

Диаграммы последовательностей (sequence diagram) — это диаграммы взаимодействия, в которых подробно описывается, как выполняются операции. Они отражают потоки событий, происходящие в рамках одного варианта использования и фиксируют взаимодействие между объектами в контексте сотрудничества.

Диаграмма активности во многом схожа с диаграммой состояний, поскольку на ней также присутствуют обозначения состояний и переходов. Диаграмма активности позволяет моделировать сложные жизненные циклы объекта с переходами из одного состояния в другое.

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности,

вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

Во **втором** разделе представлена практическая часть построения диаграмм SADT и UML. Рассматривается прикладная область работы, а именно процессы, которые автоматизирует информационная система. С помощью нотации SADT строятся модели “As Is” и “To Be”.

Функциональные модели “As Is” и “To Be” – это модели процессов в текущий момент и в будущий. “As Is” дословно переводится «Как есть сейчас». Данная функциональная модель нужна для того, чтобы проанализировать, как работает система или бизнес-процесс в настоящем. С помощью этой диаграммы можно понять, где находятся слабые места текущего процесса, а также в чем будут состоять преимущества нового. Модель “To Be” переводится как «Так должно быть».

В первую очередь, сайт театра предназначен для оптимизации продажи билетов - а именно, для его автоматизации, поэтому на моделях отражен процесс продажи билетов до автоматизации и после.

Построены диаграммы UML, которые отражают набор взаимодействий между действующим лицом и системой - с сайтом театра. На диаграмме вариантов использования приведены процессы, которые пользователь может совершать относительно системы. Действующими лицами в данной диаграмме выступают Клиент (посетитель сайта) и Администратор сайта. Пользователь может инициировать следующие действия: посмотреть сайт, афишу, выбрать место, обратиться в службу поддержки, оставить отзыв. Для всех процессов кроме просмотра сайта и афиши необходима авторизация, поэтому она является включением в эти варианты использования. Вследствие просмотра афиши пользователь может захотеть выбрать спектакль, поэтому это действие является расширением просмотра афиши.

На диаграмме последовательности представлен процесс покупки билетов. Клиент может зайти на сайт, указать фильтры поиска спектакля, купить билет. В ответ на его действия система управления передает запросы базе данных театра, экрану, банку. При оплате происходит взаимодействие между банком и счетом клиента. При покупке билета система управления оповещает клиента об успешной оплате и уведомляет базу данных, чтобы

она внесла изменения.

На диаграмме активности отображен аналогичный процесс покупки билета.

В **третьем** разделе представлена реализация сайта театра, написанного на Django Framework. Django — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

Для создания базы данных (и всех взаимодействий с ней) Django использует ORM. ORM (Object-Relational Mapping или Объектно-Реляционное Представление) в Django framework - технология программирования, которая связывает отношения базы данных с классами ООП. ORM позволяет работать с базой данных, используя объекты Python, вместо написания прямых SQL-запросов. По умолчанию Django сконфигурирован для работы с базой данных SQLite.

Django работает по модели MVT — Model-View-Template, или «Модель–Представление–Шаблон». Она разделяет внутреннюю логику работы сайта, внешний вид страниц для пользователя и реакции веб-сервиса на внешние воздействия.

Модель обеспечивает внутреннюю работу сайта: подключение к базе данных, формат хранения информации и другие функции. Представление определяет данные, которые нужно показать пользователю, и отправляет их туда, где они должны быть показаны. Также оно принимает запросы пользователей и реагирует на них — например, обновляет страницу после отправки данных формы. Шаблон — это каркас страницы на Django, которую увидит пользователь (HTML страница).

В данном разделе описывается структура проекта, из каких приложений состоит сайт. Для реализации сайта театра были созданы следующие приложения: main, посвященное главной странице, news - новостям, performances - спектаклям, афишам, актерам, билетам, корзине, users - приложение для работы с пользователями.

Структура приложения выглядит следующим образом:

— папка migrations: предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с определением моделей;

— init.py: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета;
— admin.py: предназначен для административных функций, в частности, здесь призводится регистрация моделей, которые используются в интерфейсе администратора;
— apps.py: определяет конфигурацию приложения;
— models.py: хранит определение моделей, которые описывают используемые в приложении данные;
— tests.py: хранит тесты приложения;
— urls.py: содержит url адреса приложения;
— views.py: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ.

Описываются этапы реализации авторизации и регистрации пользователя на сайте. Для данной разработки в Django существует встроенный класс “User”, который представляет собой модель пользователя. В ней присутствует базовая функциональность для работы с учетными записями пользователей. Далее создается контроллер, который будет работать как на GET запрос (когда сервер возвращает пользователю страницу авторизации), так и на POST запрос, в котором будут обрабатываться введенные пользователем данные.

Для реализации контроллера создается форма, представленная в виде класса “UserLoginForm”. В классе “Meta” указано с какой моделью и с какими полями будет работать форма. Функция-представление “login” связывает модель “User” с HTML шаблоном. В форму передаются данные пользователя, взятые из POST запроса, затем форма проверяется на валидность (на длину email или корректность данных). Если она валидна - идет обращение к логину (username) и паролю. Метод authenticate проверяет пользователя с введенными данными в базе данных, если он существует в системе - пользователь авторизуется с помощью метода login и его перенаправляют на главную страницу.

Переменная “form” добавляется в контекст. Контекст в Django — это компонент системы шаблонов, который отвечает за передачу данных из представлений в шаблоны. Он представляет собой словарь Python, содержащий пары ключ-значение, где ключи представляют имена переменных, а значения

— данные, связанные с этими переменными.

Поля, в которые пользователь вводит данные, должны быть активными, то есть обрабатывать POST запросы.

В данном разделе описывается процесс добавления билета в корзину. Первым этапом реализации покупки билета является создание афиши - перечня спектаклей, билеты на которые можно приобрести. Для этого необходим класс “Performances”, который отвечает за информацию о спектаклях, и функция-представление для их отображения.

Каждое место в зрительном зале на странице выбора мест представляет собой объект класса “Seats”. При нажатии на выбранное место вызывается функция “ticket_add”, которая принимает на вход id спектакля и id места, в результате чего создается билет. Если пользователь нажмет “Купить” вызовется функция “basket_add” и билет добавится в корзину. Тег *@login_required* означает, что добавить товар в корзину может только авторизованный пользователь. Если пользователь авторизован не был, его перенаправит на страницу для авторизации. В функции “basket_add” создается объект класса “Basket”, который содержит сведения о пользователе и о выбранным им билете. После этого выполняется проверка: если билет не был добавлен в корзину данным пользователем ранее, то он создастся и добавится в корзину.

Страница спектакля должна отображать основные сведения о спектакле, об актерах, участвующих в спектакле, их роли и фотографии спектакля. Для того чтобы реализовать данный функционал, необходимо создать класс актеров, содержащий сведения об актерах, класс занятости актера в театре - в нем будут указаны актер, спектакль и роль актера в спектакле. Для реализации фотоальбома необходимы классы “ImageAlbum” - класс альбома, и “Image” - непосредственно изображения. В класс “Image” можно загружать фотографии и добавлять их в конкретный альбом.

Описывается процесс создания страницы для оставления отзыва. Создается класс “Response”. Данный функционал создан для того, чтобы любой авторизованный пользователь мог поделиться своим впечатлением о театре. По аналогии с реализацией регистрации создается форма, полученная с POST запроса, которая проверяется на валидность, после чего она сохраня-

ется. Администратор в админ-панеле сможет увидеть отзывы посетителей.

В **заключении** приведены достигнутые результаты бакалаврской работы.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

1. Определены основные понятия методологии SADT, необходимые для построения диаграмм IDEF0.
2. Определены и проанализированы автоматизируемые бизнес-процессы.
3. Построена диаграмма IDEF0, отражающая автоматизируемые процессы, и диграммы UML, в которых представлены способы взаимодействия пользователя с системой.
4. Изучен фреймворк Django, а именно структура проекта, технология программирования ORM, шаблон проектирования MVC.
5. Разработана информационная система “Театр”, представляющая собой сайт. На сайте реализованы авторизация и регистрация пользователей, выбор мест на конкретный спектакль и добавление их в корзину, афиша театра, страницы спектаклей и актеров, фотогалерея, страница для оставления отзывов. Описаны значимые модули, классы, логика построения.