2 Проектирование задачи

2.1 Моделирование проекта

Диаграммы вариантов использования[4] предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы. Другими словами, диаграммы вариантов использования говорят о том, что система должна делать, не указывая применяемые методы.

Диаграмма вариантов использования, представлена на рисунке 2.1.

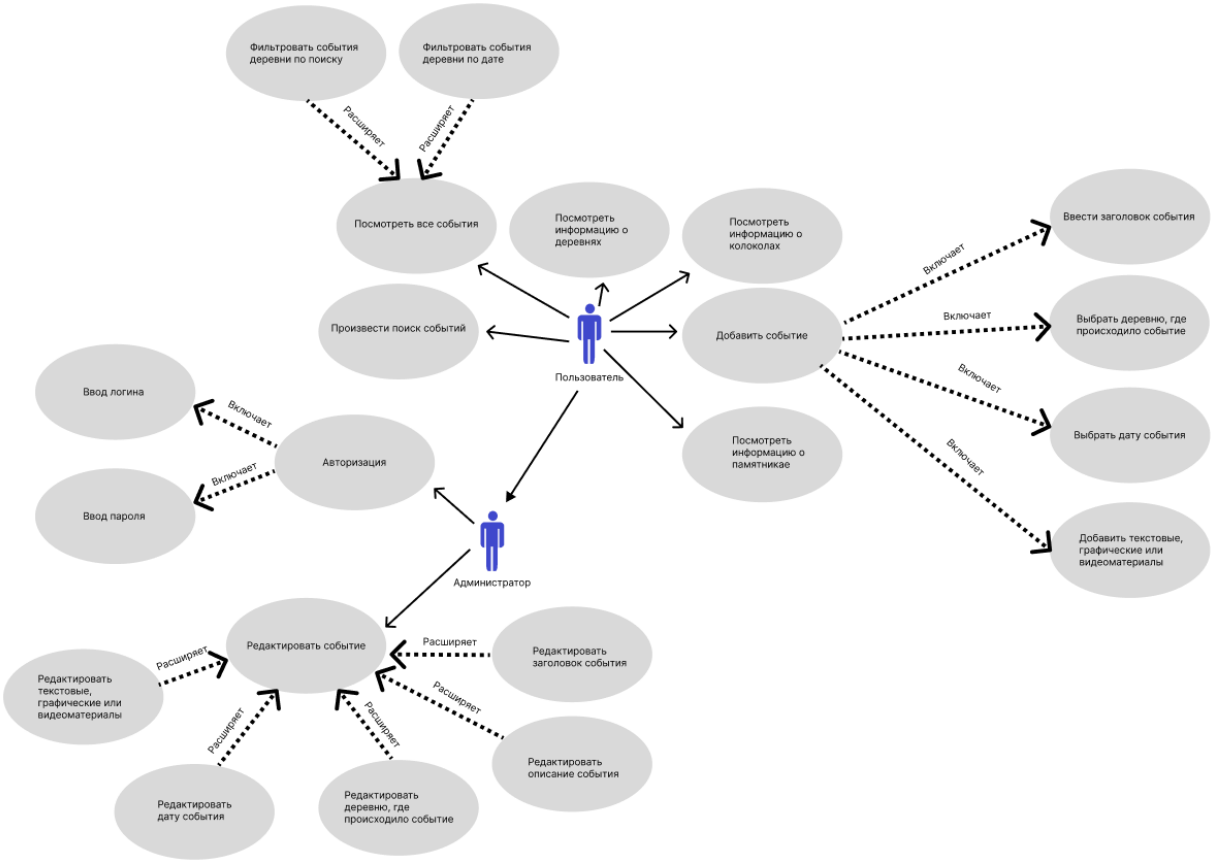

Рисунок 2.1 – диаграмма вариантов использования
Источник: собственная разработка

В диаграмме вариантов использования, представленной на рисунке 2.1, отображен функционал проекта, показывающий возможности пользователя и администратора сайта.

У пользователя есть возможность просмотра и добавления событий, просмотра информации о деревнях, о колоколах в деревне Хатынь и о памятнике «Непокоренный человек».

У администратора присутствуют все те же возможности что и у пользователя, однако также есть возможность авторизации в панели администратора. После авторизации в панели отображаются списки записей, содержащие в себе информацию. Администратор имеет право редактировать каждое событие, а именно заголовок, описание, добавленные графические материалы и так далее.

Диаграммы деятельности[5] используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Деятельность может содержать входящие и/или исходящие дуги, показывающие потоки управления и потоки данных. Если поток соединяет две деятельности, то это поток управления.

Для создания диаграммы деятельности используются следующие узлы:

узел управления (control node) – это абстрактный узел действия, координирующий потоки действий;

начальный узел деятельности (или начальное состояние деятельности) (activity initial node) является узлом управления, начинающим поток (или потоки) при вызове данной деятельности извне;

конечный узел деятельности (или конечное состояние деятельности) (activity final node) является узлом управления, останавливающим (stop) все потоки данной диаграммы деятельности, на диаграмме может быть более одного конечного узла;

конечный узел потока (или конечное состояние потока) (flow final node) является узлом управления, завершающим данный поток, на другие потоки и деятельность данной диаграммы это не влияет;

Точка разделения обеспечивает разделение одного потока на несколько параллельных потоков.

Точка слияния обеспечивает синхронизацию нескольких параллельных потоков.

У пользователя и администратора разные средства для взаимодействия с сайтом, поэтому на рисунке 2.2 и 2.3 приведены диаграммы деятельности для пользователя и администратора соответственно.

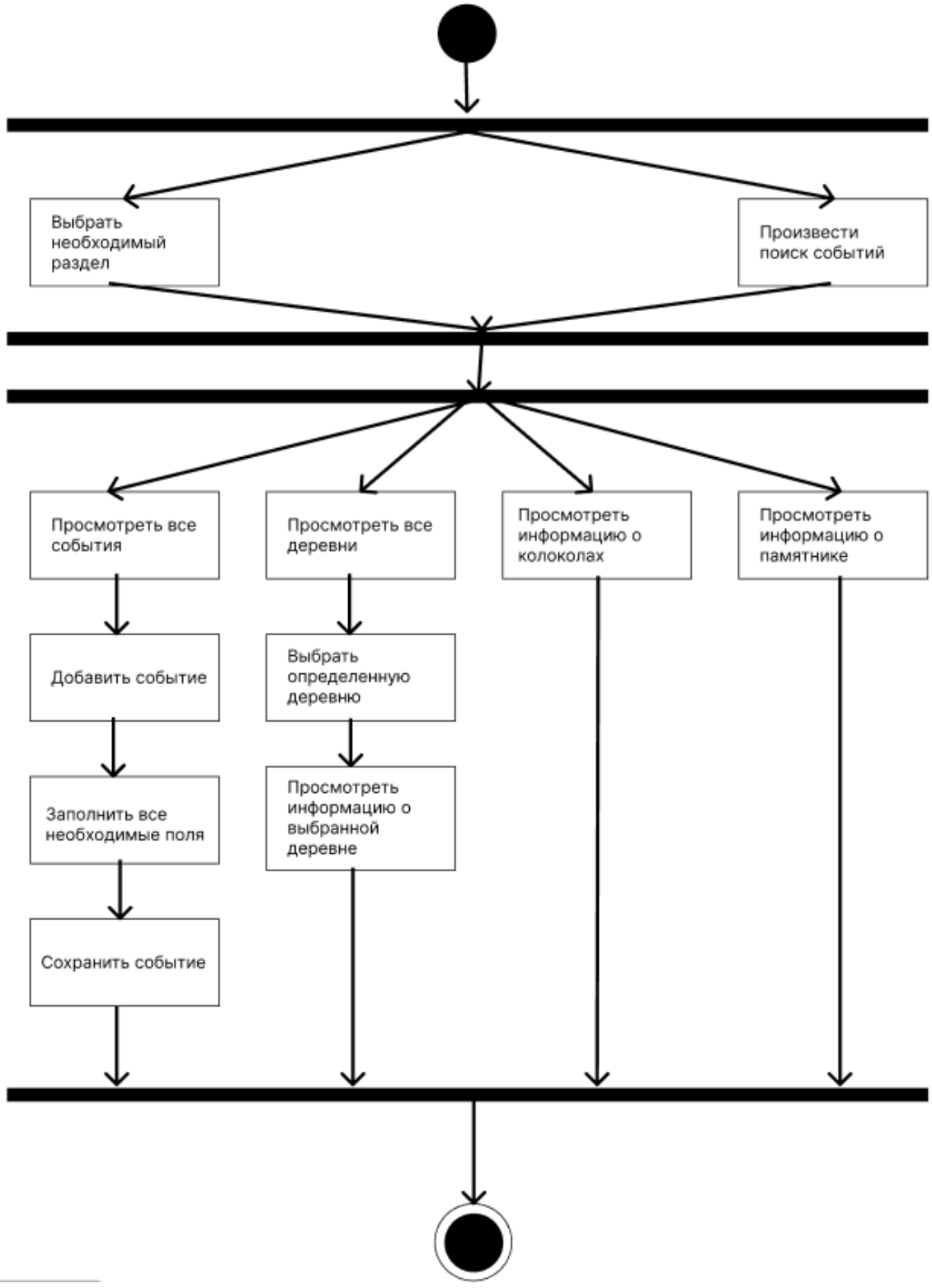

Рисунок 2.2 – диаграмма деятельности для пользователя
Источник: собственная разработка

Рисунок 2.3 – диаграмма деятельности для администратора
Источник: собственная разработка

В диаграмме, представленной на рисунке 2.2 отображается алгоритм работы пользователя.

Пользователь может найти событие с помощью строки поиска или выбрать интересующий его раздел. После выбора раздела присутствуют такие возможности как просмотр определенной информации или создание нового

события. При создании нового события пользователь должен заполнить все необходимы поля такие как:

заголовок;

деревня, где происходило событие;

дата события.

После заполнения пользователь сохраняет событие, далее запись будет проверена администратором.

Диаграмма, представленная на рисунке 2.3 отображает алгоритм работы администратора.

После авторизации для администратора отображаются определенные таблицы, содержащие в себе разнообразные записи. После выбора необходимой таблицы появляется список записей. При выборе определенной записи администратор может просмотреть подробную информацию о событии или достопримечательности. При необходимости администратор может отредактировать и сохранить запись.

Диаграмма классов[6] – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперацией, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между объектами. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Основными элементами диаграммы классов являются классы, также в диаграмме указываются связи между объектами.

Диаграмма классов представлена на рисунке 2.4.

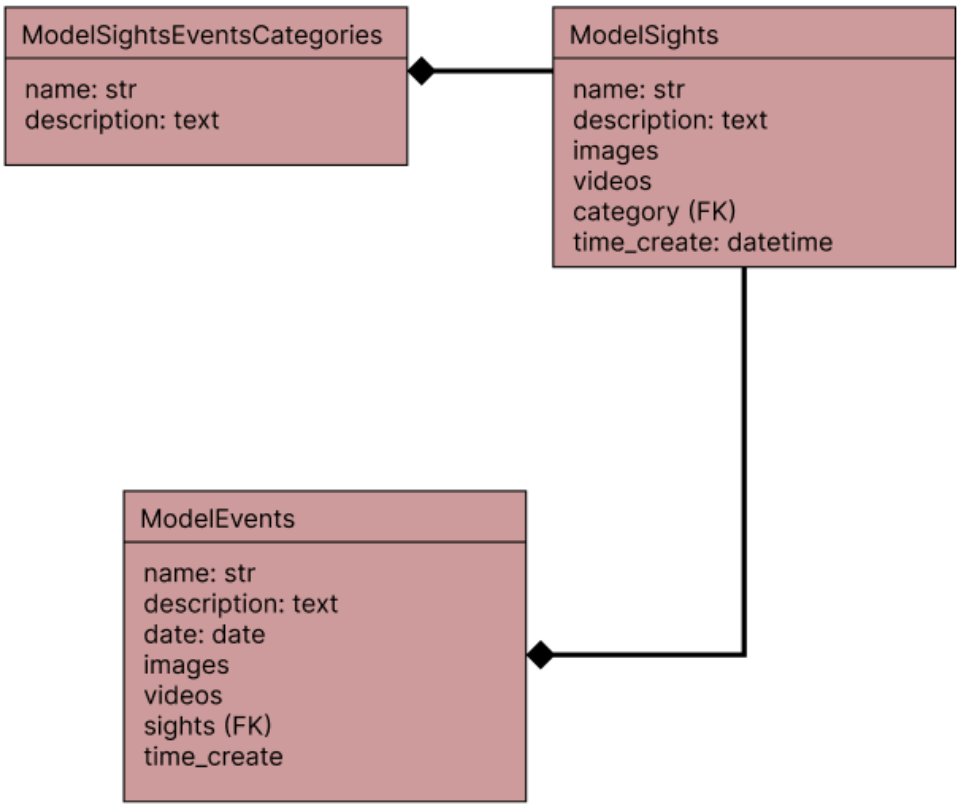

Рисунок 2.4 – диаграмма классов
Источник: собственная разработка

Диаграмма классов, представленная на рисунке 2.4 отображает все объекты базы данных, необходимые для корректной работы веб-сайта.

Объект «ModelSightsEventsCategories» содержит такие атрибуты как наименование и описание.

Объект «ModelSights» содержит такие атрибуты как:

наименование;

описание;

атрибут для хранения изображения;

атрибут для хранения видеоизображения;

внешний ключ, ссылающийся на объект «ModelSightsEventsCategories»;

время создания объекта.

Объект «ModelEvents» содержит такие атрибуты как:

наименование;

описание;

дата события;

атрибут для хранения изображения;

атрибут для хранения видеоизображения;

внешний ключ, ссылающийся на объект «ModelSights»;

время создания объекта.

Так как в разрабатываемом проекте присутствует база данных, то на рисунке 2.5 отображена структура базы данных в виде диаграммы сущность-связь[7].

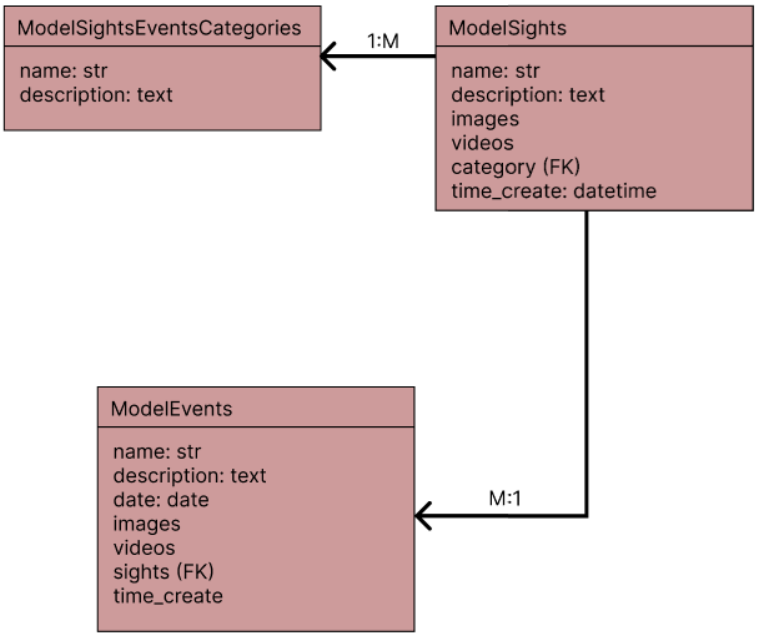

Рисунок 2.5 – диаграмма сущность-связь
Источник: собственная разработка

Диаграмма сущность-связь отображает такую же информацию что и диаграмма классов, за исключением отображения типа связи.

На один объект «ModelSightsEventsCategories» может приходиться несколько объектов «ModelSights», также как на один объект «ModelSights» может приходиться несколько объектов «ModelEvents».

2.2 Описание системы меню

Системы меню для администратора и пользователя имеют существенные отличия. Администратор зачастую работает через специальную панель администратора, в то время как у пользователя присутствует полноценная система меню на главной странице веб-сайта.

Система меню, разработанная для администратора и для пользователя представлены на рисунке 2.6 и 2.7 соответственно

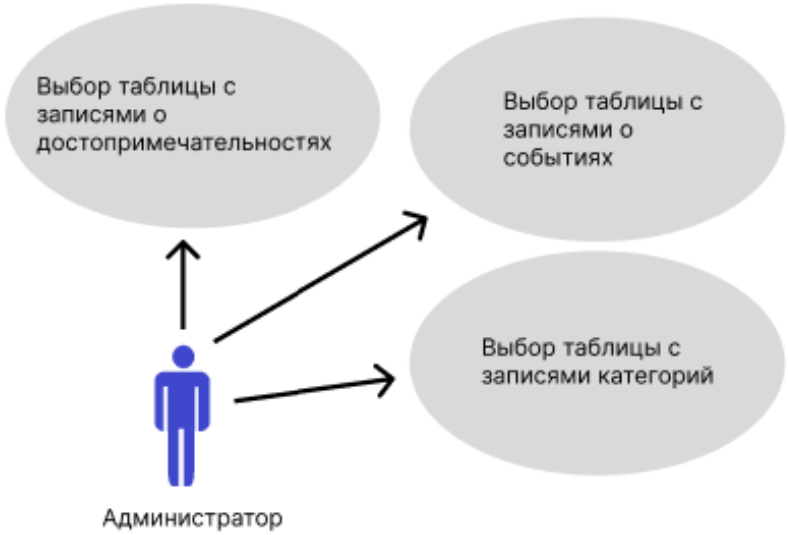

Рисунок 2.6 – схема системы меню администратора
Источник: собственная разработка

Администратор работает через специальную панель, схожую на СУБД. В панели отображены таблицы, созданные с помощью классов моделей, написанных на языке программирования Python. Также в панели отображаются таблицы с пользователями и таблицы, добавленные посредством установки разнообразных расширений для Django. С помощью панели у администратора есть возможность редактирования определенных полей записей, также все записи отображены в виде удобного списка. При необходимости панель администратора дополняется разнообразными фильтрами и дополнительными модулями.

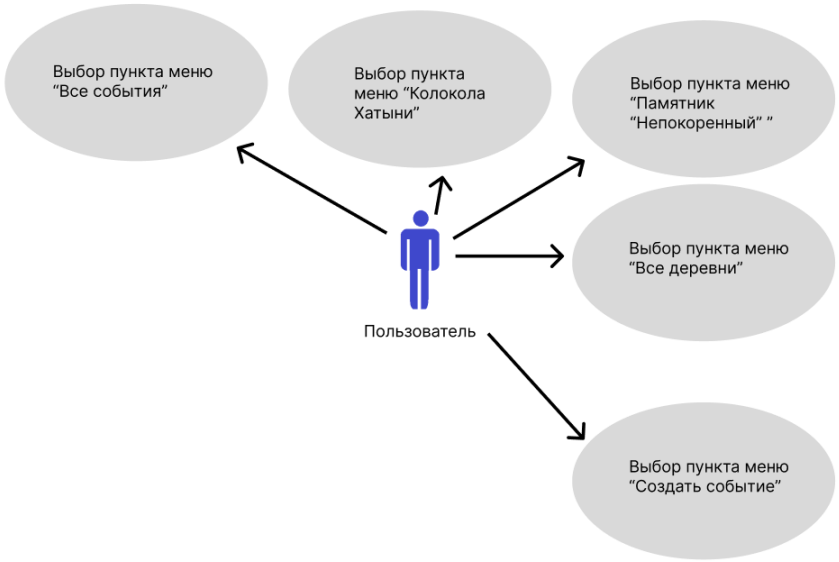

Рисунок 2.7 – схема системы меню пользователя
Источник: собственная разработка

Система меню для пользователя находится на главной странице веб-сайта и представляет собой стену памяти, расположенную в мемориальном комплексе «Хатынь». На стене вместо надписей находятся ссылки на другие страницы сайта. При наведении на ссылку изменяется цвет текста, при клике пользователь переходит на выбранную страницу.

2.3 Выбор и обоснование среды разработки

Python – высокоуровневый язык программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости и качества кода, а также на обеспечение переносимости написанных на данном языке программирования программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью Python является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками являются зачастую более низкая скорость работы и более высокое потребление памяти написанных программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Python является мультипарадигменным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, метапрограммирование и функциональное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Аспектно-ориентированное программирование частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными фреймворками. Такие методики как контрактное и логическое программирование можно реализовать с помощью библиотек или расширений.

Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Для разработки проекта использовался фреймворк «Django».

Django – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC – «Модель-представление-контроллер».

Первоначально разработка Django велась для обеспечения более удобной работы с новостными ресурсами, что достаточно сильно отразилось на архитектуре: фреймворк предоставляет ряд средств, помогающих в быстрой разработке веб-сайтов информационного характера. Например, разработчику не требуется создавать контроллеры и страницы для административной части сайта,

в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка – DRY.

Django предоставляет следующие возможности:

ORM, API доступа к БД с поддержкой транзакций;

встроенный интерфейс администратора, с уже имеющимися переводами на многие языки;

диспетчер URL на основе регулярных выражений;

расширяемая система шаблонов с тегами и наследованием;

система кеширования;

подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты;

«generic views» — шаблоны функций контроллеров;

авторизация и аутентификация, подключение внешних модулей аутентификации: LDAP, OpenID и прочие.;

библиотека для работы с формами (наследование, построение форм по существующей модели БД).