Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

НОВОСТНОЙ САЙТ

Выполнил: Брычиков Д. Д.
студент гр.553501

Проверил: Алексеев Ю. И.

Минск 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире через каждого человека проходят большие потоки информации. Не всегда человеку легко выделить из этого потока ключевую информацию. Однако каждый человек хочет быть информированным о ключевых событиях, происходящим в нашем мире.

Основываясь на данной потребности человека, в интернете начали возникать всякого рода новостные порталы. Посетив их, человек может в кратчайшие сроки получить перечень кратки перечень произошедших событий и получить более подробную информацию о событиях, которые больше всего интересуют человека.

Новостные порталы могут иметь самую разнообразную тематику. В настоящее время почти на каждую сферу можно найти тематический сайт. Однако также следует отметить, что количество тем для таких порталов фактически ничем не ограниченно. В мире постоянно появляются новые идейные течения, которые можно организовать в области. Следовательно, и новостные порталы различной тематики могут появляться постоянно.

По источнику информации новостные могут различаться. Крупные сайты имеют своих журналистов, которые оперативно реагирую на события, проводят расследование и пишут об этом статьи. Некоторые сайты сотрудничают с государственными структурами, которые через представителей новостных порталов доносят информацию до остальных людей. Однако большинство новостных сайтов являются лишь агрегатами новостей с других порталов.

Таким образом практически любой человек может открыть свой новостной сайт. Пользуясь другими источниками информации, пользователь может агрегировать информацию с других авторитетных источников, и публиковать те новости, которые автор сочтет интересным для его публики.

Однако существует техническая сложность в реализации платформы для публикации новостей. Именно о технической реализации гипотетического новостного сайта и пойдет речь в данной работе.

1 ПОСТАНОВКА ЗАДАЧИ

В качестве объекта исследования были выбраны новости.

Предметом исследования являются компьютерная платформа для доставки новостей пользователям.

Целью исследования создание гипотетического новостного портала.

Для достижения цели перед началом работы был поставлен ряд задач:

– исследование феномена новостей
– изучение необходимой технической информации
– выявление потребностей у поставщиков и потребителей новостей
– синтез структуры новостной платформы
– реализация новостной платформы

2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Феномен новостей

Новости — оперативная информация, которая представляет политический, социальный или экономический интерес для аудитории в своей свежести, то есть сообщения о событиях, произошедших недавно или происходящих в данный момент. Также новостями называют программы (собрание нескольких новостей) на телевидении и радио, а в печатной прессе или на веб-сайтах — сводки новостей, в специальной рубрике в газете.

2.1.1 История новостей

Издавна новости распространялись, зачастую в искажённом виде, как слухи, от человека к человеку. Официальные новости в древности доставляли гонцы, объявляли их для всеобщего сведения глашатаи. В Древнем Риме богатые люди, жившие в колониях, имели в Риме своих личных корреспондентов-хроникёров, которые должны были держать их в курсе всего происходящего в столице. Большинство этих корреспондентов были образованными рабами. Например, когда Цицерон был проконсулом, то некий Хрест сообщал ему из Рима политические новости, пересылал отчёты о гладиаторских боях, городских происшествиях и ходивших по городу сплетнях.

Переписывавшиеся от руки свитки под названием Acta diurna populi romani («Ежедневные дела римского народа») вывешивались на площадях и доставлялись политикам или просто знатным горожанам. Римские газеты представляли собой деревянные дощечки, на которых записывали хронику событий. Новостные сводки, как правило, имели неофициальный характер, пока Юлий Цезарь не распорядился в обязательном порядке распространять отчёты о заседаниях сената, донесения полководцев и послания правителей соседних государств.

Первой в мире печатной газетой стал «Столичный вестник», который начал выходить в Китае в VIII веке. В ней помещали указы императора и сообщения о важнейших событиях. Газеты печатали с досок, на которых вырезали иероглифы, покрывали тушью и делали оттиски. Эта технология была крайне неудобной, так как доска от частого покрывания краской быстро приходила в негодность.

В средневековой Европе необходимость поддерживать сообщения между городами заставляла правительства, учреждения и некоторых частных лиц заводить особых гонцов, которые постоянно ездили между определёнными городами, передавая от одного в другой различного рода сообщения. В конце XV века для поддержания постоянного обмена известиями между различными пунктами правительственные учреждения, монастыри, князья, университеты начали усиленно пользоваться такими гонцами и между наиболее центральными и оживлёнными пунктами

установился чрезвычайно деятельный и вполне урегулированный обмен известиями. Сначала известия эти были не чем иным, как сообщениями частных лиц частным лицам, или же правительственными циркулярами. Но постепенно всё больший круг лиц начинал интересоваться всевозможными вестями, привозимыми гонцами, они стали распространять среди лиц, для которых они представляли уже не личный, а общественный интерес. Письма, адресованные частному лицу, но представлявшие общий интерес, начали переписываться в нескольких экземплярах и рассылаться знакомым. Таким путём частная переписка постепенно развивалась в общественную рукописную газету.

Вплоть до изобретения в Германии в 1450-х годах Иоганном Гутенбергом печатного пресса, позволявшего размножать текст и изображения, не прибегая к услугам переписчиков, газеты (представлявшие собой всё те же переписанные от руки свитки с главными новостями) оставались весьма дорогим атрибутом жизни высокопоставленных чиновников или богатых торговцев. Свой современный облик газеты начали приобретать в XVI веке. Тогда и вошло в обиход само название «газета» — по наименованию мелкой итальянской монеты газетты, которую платили за листок новостей вен. La gazeta dele novità (буквально «Новостей на газетту») в Венеции. Считается, что именно в этом городе были образованы первые бюро по сбору информации — прообразы информационных агентств — и возникла профессия «писателей новостей».

Получение первого телеграфного сообщения в Лондоне из Парижа в 1851 году.

В XIX веке появление электрического телеграфа сделало возможным для газет с невиданной до того быстротой доводить до сведения читателей новости о событиях во всём мире.

Появление радиовещания в 1920-е годы ещё ускорило распространение новостей.

Распространение телевидения начиная с 1940-х годов позволило получать не только текстовую или звуковую, но и видеоинформацию о событиях. К концу XX века благодаря развитию спутникового и кабельного телевидения появились специальные новостные телеканалы, вещающие круглосуточно.

Сотрудник телекомпании Al Jazeera готовится вести репортаж о хадже из Мекки.

Наконец, развитие Интернета, появление Интернет-СМИ привело к тому, что новости в настоящее время распространяются практически мгновенно по всему миру.

2.1.2 Изложение новостей

Как правило, новости на телевидении и радио передаются несколько раз в день, начинаются в начале часа и длятся от двух минут до часа. Новости обычно бывают из таких областей, как политика, экономика, наука, культура,

спорт, с прогнозом погоды в заключении. Западной традицией предполагается, что новости должны быть изложены максимально нейтрально и объективно и отдельно от комментариев. Выбор новостей для подборки осуществляется редакцией. Специальный выпуск — это выпуск, посвящённый только одной теме, продолжительность и периодичность специальных выпусков может отличаться от обычных выпусков.

Новости — это ответ на пять вопросов: кто, что, когда, где и почему (все на букву W: англ. who, what, when, where, why).

Хорошо написанная вещь — ясная, легко читаемая, в ней используется незакостеневший язык, она поучает и развлекает. Все эти характеристики подходят к хорошо написанной газетной статье в той же мере, что и к хорошо написанному роману.

Анализируя ситуацию в медиа-индустрии, Том Вулф резонно отмечал, что в новой журналистике использованы типичные литературные приёмы:

- Сценоописательство.
- Диалоги героев материала, использование разговорной «живой» речи.
- Отчётливо проговорённая в публикации личная точка зрения автора, употребление местоимения «я», интерпретация описываемых событий с точек зрения конкретных персонажей.
- Литературная регистрация каждодневных деталей (поведения и т. п.) персонажей, подробная фиксация т. н. статусов действующих лиц.

Способ подачи материала влияет на конструкцию, основные контуры которой необходимо представить до того, как начинать писать. В простом информационном сообщении план не займет много времени и сложится в уме. Но для длинных или сложных статей его лучше набросать на бумаге. План вовсе не обязательно делать подробным — достаточно расположить по полочкам основные блоки, возможно, с небольшими пометками о том, как эти блоки связать.

Существуют рекомендации, которым должны следовать журналисты, работающие в массовых СМИ:

1. Добиваться ясности ещё до того, как напишете хоть слово.
2. Как можно осторожнее включайте каждый новый этап в своё повествование, каждое событие в цепь событий, каждый довод в доказательство.
3. Не считайте, что читатели наперёд знают всё и разбираются во всем.
4. Объясняйте жаргон. Представители крупных компаний говорят, что их «операционная единица» (то есть, их фирма) «переживает некоторые затруднения с потоком наличности» (то есть, сидит без денег) из-за «проблем с внедрением на рынок» (иными словами, никто не покупает их продукцию), так что необходима «рационализация производственного процесса» (читай: будут увольнять работников). Правительственные чиновники говорят об «исправительных учреждениях», имея в виду тюрьмы, и об «отсутствии баланса между

предложением жилых помещений и спросом на них», подразумевая нехватку жилья.
5. Убедитесь, что написанные фразы предельно ясны.
6. Избегайте мудрёной манеры письма и заумного языка.
7. Последнее слово на тему ясности — простота[1].

2.2 Базы данных

Для реализации новостного портала ключевую роль играет способ хранения информации.

База данных — представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

Многие специалисты указывают на распространённую ошибку, состоящую в некорректном использовании термина «база данных» вместо термина «система управления базами данных», и указывают на необходимость различения этих понятий.

2.2.1 Виды баз данных

Существует огромное количество разновидностей баз данных, отличающихся по различным критериям. Например, в «Энциклопедии технологий баз данных», по материалам которой написан данный раздел, определяются свыше 50 видов БД.

Основные классификации приведены ниже.

Классификация по модели данных:

- Иерархическая
- Объектная и объектно-ориентированная
- Объектно-реляционная
- Реляционная
- Сетевая
- Функциональная.

Классификация по среде постоянного хранения:

- Во вторичной памяти, или традиционная (англ. conventional database): средой постоянного хранения является периферийная энергонезависимая память (вторичная память) — как правило жёсткий диск. В оперативную память СУБД помещает лишь кэш и данные для текущей обработки.
- В оперативной памяти (англ. in-memory database, memory-resident database, main memory database): все данные на стадии исполнения находятся в оперативной памяти.

- В третичной памяти (англ. tertiary database): средой постоянного хранения является отсоединяемое от сервера устройство массового хранения (третичная память), как правило на основе магнитных лент или оптических дисков. Во вторичной памяти сервера хранится лишь каталог данных третичной памяти, файловый кэш и данные для текущей обработки; загрузка же самих данных требует специальной процедуры.

Классификация по содержимому:

- Географическая
- Историческая
- Научная
- Мультимедийная
- Клиентская.

Классификация по степени распределённости:

- Централизованная, или сосредоточенная (англ. centralized database): БД, полностью поддерживаемая на одном компьютере.
- Распределённая БД (англ. distributed database) — составные части которой размещаются в различных узлах компьютерной сети в соответствии с каким-либо критерием.

Другие виды БД

- Пространственная (англ. spatial database): БД, в которой поддерживаются пространственные свойства сущностей предметной области. Такие БД широко используются в геоинформационных системах.
- Временная, или темпоральная (англ. temporal database): БД, в которой поддерживается какой-либо аспект времени, не считая времени, определяемого пользователем.
- Пространственно-временная (англ. spatial-temporal database) БД: БД, в которой одновременно поддерживается одно или более измерений в аспектах как пространства, так и времени.
- Циклическая (англ. round-robin database): БД, объём хранимых данных которой не меняется со временем, поскольку в процессе сохранения новых данных они заменяют более старые данные. Одни и те же ячейки для данных используются циклически[2].

2.2.2 Нормализация

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации. Как отмечает К. Дейт, общее назначение процесса нормализации заключается в следующем:

- исключение некоторых типов избыточности;
- устранение некоторых аномалий обновления;
- разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
- упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

2.2.3 Роль нормализации в проектировании реляционных баз данных

При том, что идеи нормализации весьма полезны для проектирования баз данных, они отнюдь не являются универсальным или исчерпывающим средством повышения качества проекта БД. Это связано с тем, что существует слишком большое разнообразие возможных ошибок и недостатков в структуре БД, которые нормализацией не устраняются. Несмотря на эти рассуждения, теория нормализации является очень ценным достижением реляционной теории и практики, поскольку она даёт научно-строгие и обоснованные критерии качества проекта БД и формальные методы для усовершенствования этого качества. Этим теория нормализации резко выделяется на фоне чисто эмпирических подходов к проектированию, которые предлагаются в других моделях данных. Более того, можно утверждать, что во всей сфере информационных технологий практически отсутствуют методы оценки и улучшения проектных решений, сопоставимые с теорией нормализации реляционных баз данных по уровню формальной строгости.

Нормализацию иногда упрекают на том основании, что «это просто здравый смысл», а любой компетентный профессионал и сам «естественным образом» спроектирует полностью нормализованную БД без необходимости применять теорию зависимостей. Однако, как указывает К. Дейт, нормализация в точности и является теми принципами здравого смысла, которыми руководствуется в своём сознании зрелый проектировщик, то есть принципы нормализации — это формализованный здравый смысл. Между тем, идентифицировать и формализовать принципы здравого смысла — весьма трудная задача, и успех в её решении является существенным достижением.

2.2.4 Нормальные формы

Первая нормальная форма (1NF)

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

Вторая нормальная форма (2NF)

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа.

Третья нормальная форма (3NF)

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

Нормальная форма Бойса — Кодда (BCNF)

Переменная отношения находится в нормальной форме Бойса — Кодда (иначе — в усиленной третьей нормальной форме) тогда и только тогда, когда каждая её нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ.

Четвёртая нормальная форма (4NF)

Переменная отношения находится в четвёртой нормальной форме, если она находится в нормальной форме Бойса — Кодда и не содержит нетривиальных многозначных зависимостей.

Пятая нормальная форма (5NF)

Переменная отношения находится в пятой нормальной форме (иначе — в проекционно-соединительной нормальной форме) тогда и только тогда, когда каждая нетривиальная зависимость соединения в ней определяется потенциальным ключом (ключами) этого отношения.

Доменно-ключевая нормальная форма (DKNF)

Переменная отношения находится в ДКНФ тогда и только тогда, когда каждое наложенное на неё ограничение является логическим следствием ограничений доменов и ограничений ключей, наложенных на данную переменную отношения.

Шестая нормальная форма (6NF)

Переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ[3].

3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

Проведя достаточно исследований предметной области, было реализована платформа для доставки новостей.

3.1 Выбор средств разработки

В качестве языка программирования был выбран python.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций[4].

В качестве фреймворка для реализации платформы был выбран Django.

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

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 (англ. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений[5].

3.2 Проектирование базы данных

Основой частью проекта является проектирование базы данных. Были выделены следующие основные сущности:

- Новость
- Комментарии
- Пользователи
- Роли пользователей

Кроме этого было введен ряд служебных таблиц для функционирования сайта.

Основные сущности были частично нормализованы до степени, необходимой для функционирования проекта.

3.3 Роли пользователей

Для реализации платформы были определены группы пользователи с правами.

Посетитель:

- список новостей
- читать новость
- читать комментарии

Обычный пользователь:

- все права посетителя
- оставлять комментарии

Администратор

- все права обычного пользователя
- создание новостей
- редактирование новостей
- удаление новостей
- удаление комментариев

3.4 Схема переходов сайта

Для портала был реализован ряд страниц со следующей схемой перехода между ними.

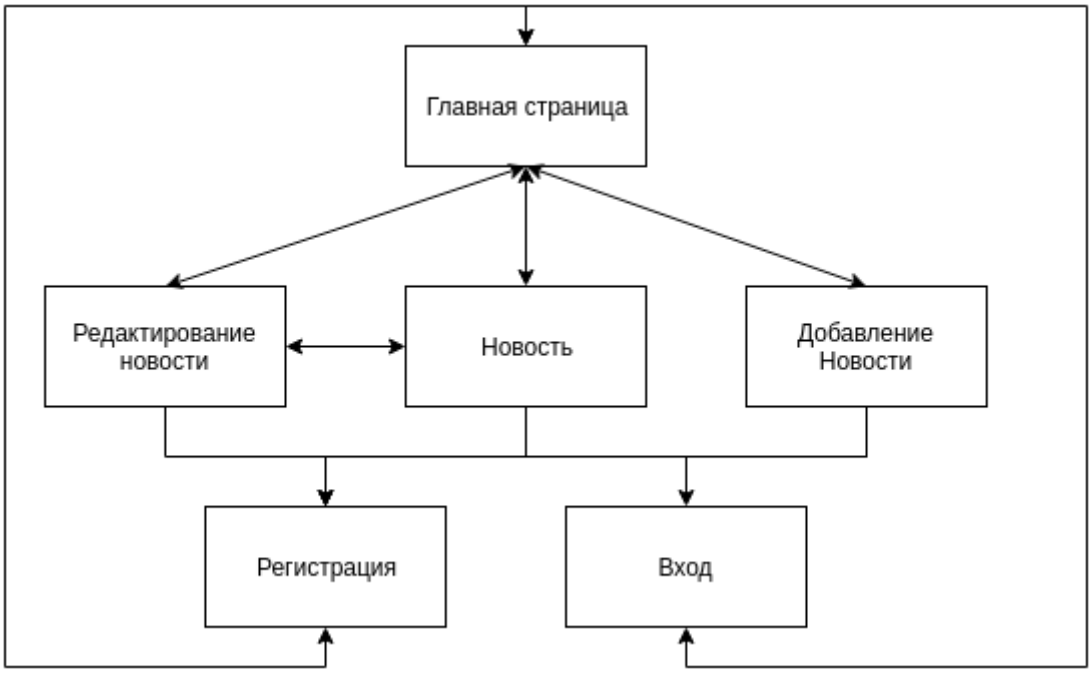


Рисунок 1. Схема переходов сайта

3.5 Страницы сайта

Для платформы были реализованы страницы с использование HTML и CSS.

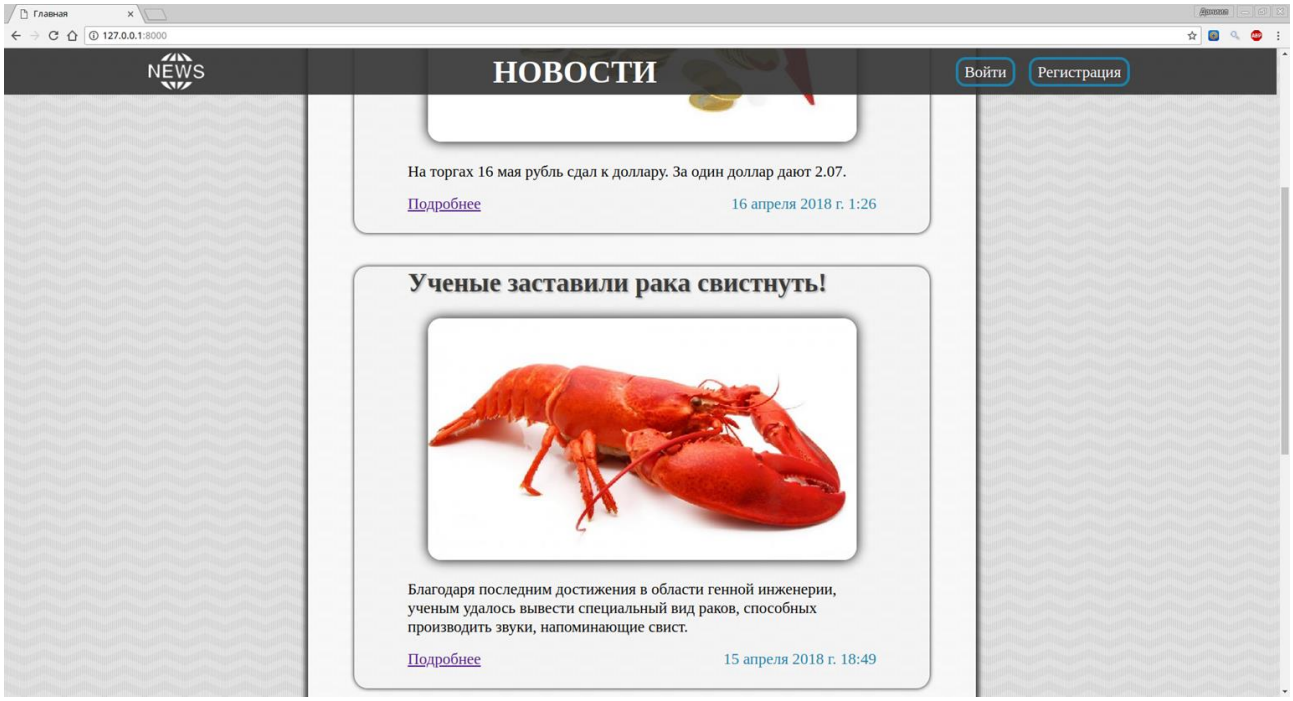


Рисунок 2. Главная страница сайта

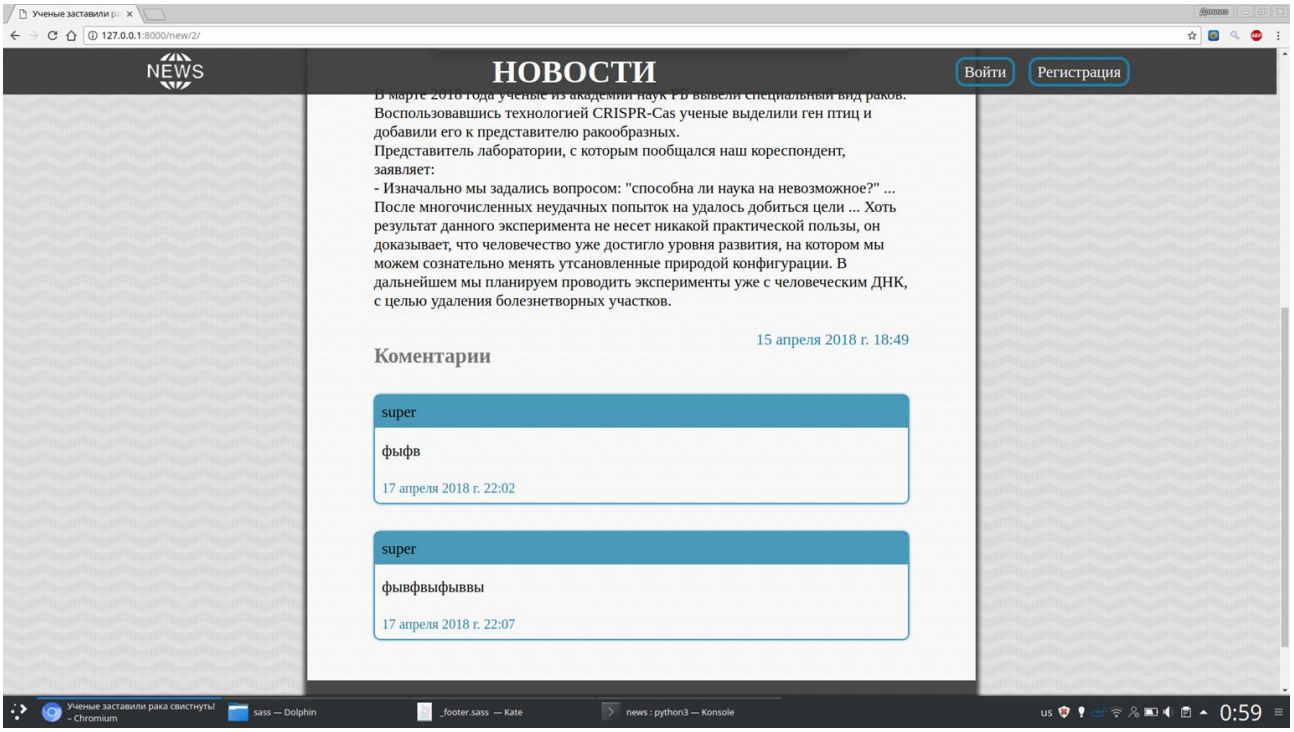


Рисунок 3. Информация о новости с комментариями

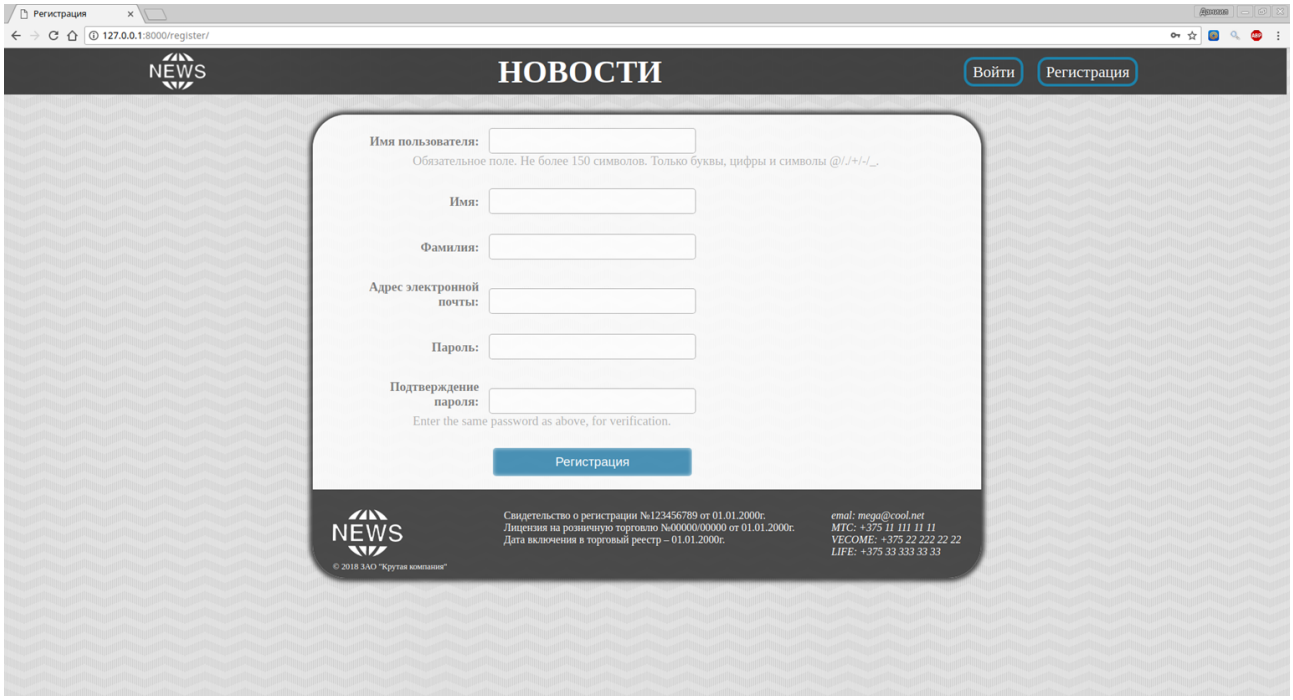


Рисунок 4. Страница регистрации

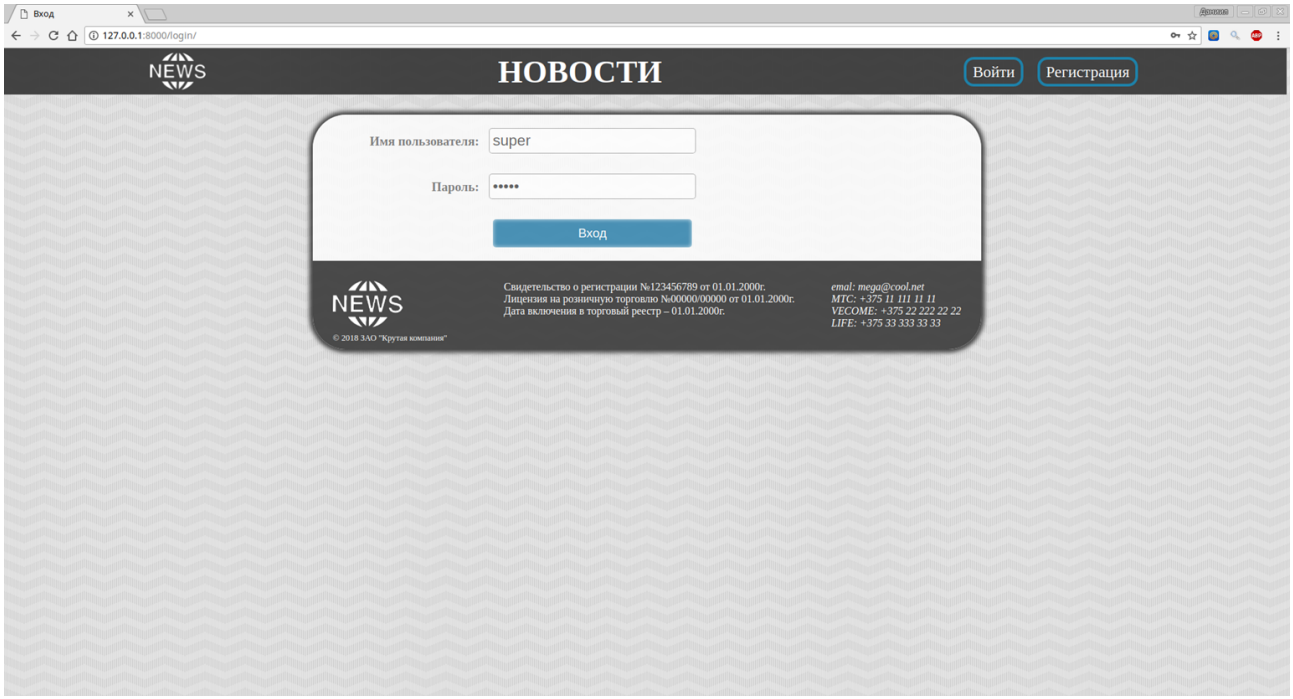


Рисунок 5. Страница входа

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был проанализирован феномен новостей.

Новости — оперативная информация, которая представляет политический, социальный или экономический интерес для аудитории в своей свежести, то есть сообщения о событиях, произошедших недавно или происходящих в данный момент.

Издавна новости распространялись, зачастую в искажённом виде, как слухи, от человека к человеку. Развитие Интернета, появление Интернет-СМИ привело к тому, что новости в настоящее время распространяются практически мгновенно по всему миру.

Для реализации платформы ля распространения новостей очень важно правильно организовать хранение информации. В этом может помочь теория базы данных.

Для обеспечения гибкости и консистентной базы данных следует использовать нормализацию. Существует несколько форм нормализации. Как правило базу доводят до 2-3 нормальной формы.

В результате работы была синтезирована структура базы данных, которая позволяет организовать полноценный новостной сайт с минимальным функционалом.

Было реализовано приложения с использованием Django и Python. Приложения является прототипом, но позволяет полноценно продемонстрировать работу гипотетической системы.

Результат проделанной работы может быть использован для реализации собственного полноценного интернет портала.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сайт Wikipedia. Новости. [Электронный ресурс]. / Режим доступа: https://ru.wikipedia.org/wiki/Новости

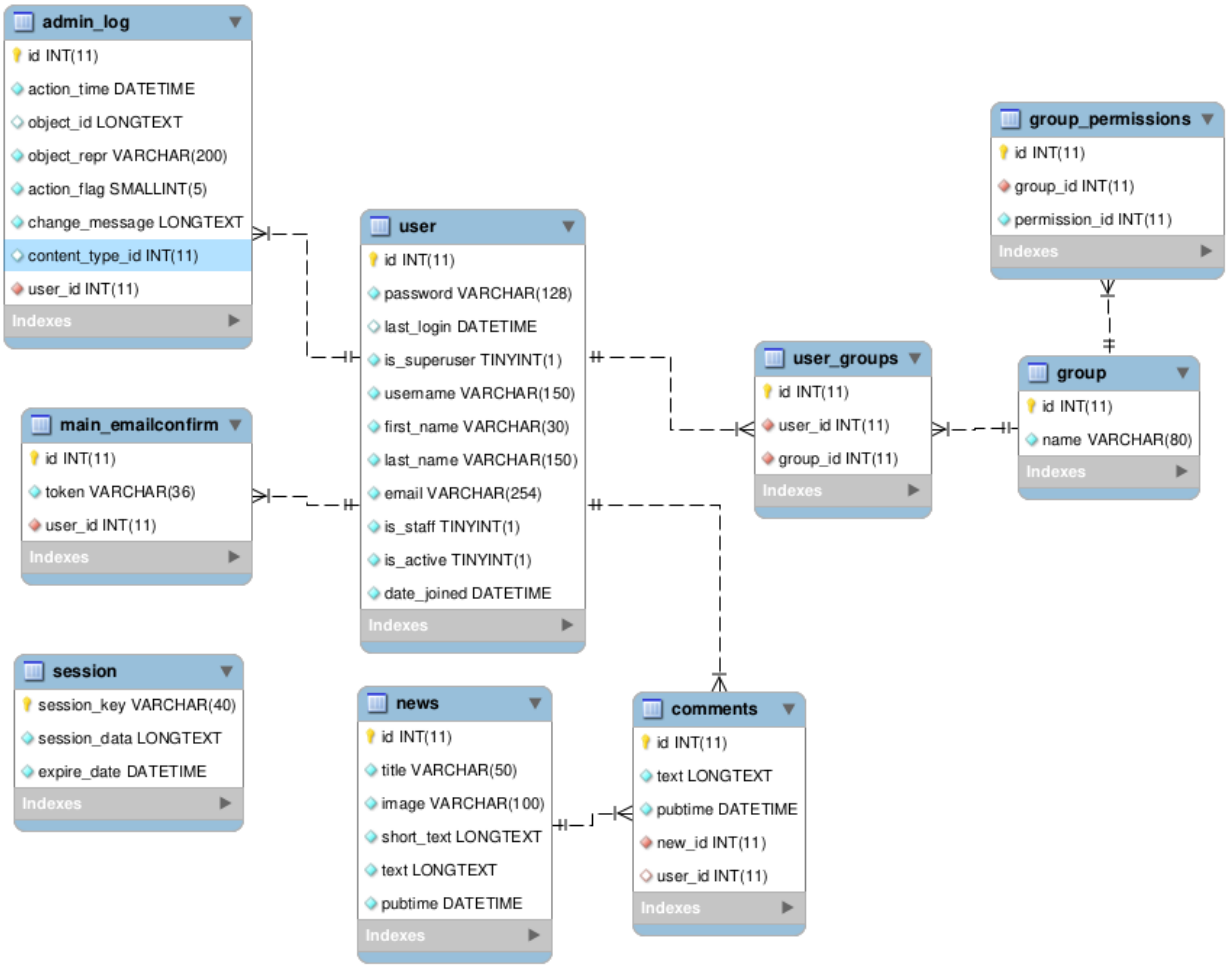
2. Сайт Wikipedia. База данных. [Электронный ресурс]. / Режим доступа: https://ru.wikipedia.org/wiki/База_данных

3. Сайт Wikipedia. Нормальная форма. [Электронный ресурс]. / Режим доступа: https://ru.wikipedia.org/wiki/Нормальная_форма

4. Сайт Wikipedia. Python. [Электронный ресурс]. / Режим доступа: https://ru.wikipedia.org/wiki/Python

5. Сайт Wikipedia. Django. [Электронный ресурс]. / Режим доступа: https://ru.wikipedia.org/wiki/Django

ПРИЛОЖЕНИЕ А. Схема базы данных



ПРИЛОЖЕНИЕ Б. Скрипт для построения базы данных

```
CREATE TABLE `NEWS`.`group` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(80) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `name`);


CREATE TABLE `NEWS`.`group_permissions` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `group_id` INT(11) NOT NULL,
  `permission_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `group_permissions_id_uniq`,
  CONSTRAINT `fk_auth_group_id`
    FOREIGN KEY (`group_id`)
    REFERENCES `NEWS`.`group` (`id`));


CREATE TABLE `NEWS`.`user` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `password` VARCHAR(128) NOT NULL,
  `last_login` DATETIME NULL DEFAULT NULL,
  `is_superuser` TINYINT(1) NOT NULL,
  `username` VARCHAR(150) NOT NULL,
  `first_name` VARCHAR(30) NOT NULL,
  `last_name` VARCHAR(150) NOT NULL,
  `email` VARCHAR(254) NOT NULL,
  `is_staff` TINYINT(1) NOT NULL,
  `is_active` TINYINT(1) NOT NULL,
  `date_joined` DATETIME NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username`);


CREATE TABLE `NEWS`.`user_groups` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `user_id` INT(11) NOT NULL,
  `group_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `user_groups_id_uniq`,
  CONSTRAINT `fk_group_id`
    FOREIGN KEY (`group_id`)
    REFERENCES `NEWS`.`group` (`id`),
  CONSTRAINT `fk_user_id`
    FOREIGN KEY (`user_id`)
    REFERENCES `NEWS`.`user` (`id`));


CREATE TABLE `NEWS`.`news` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(50) NOT NULL,
  `image` VARCHAR(100) NOT NULL,
  `short_text` LONGTEXT NOT NULL,
  `text` LONGTEXT NOT NULL,
  `pubtime` DATETIME NOT NULL,
  PRIMARY KEY (`id`));


CREATE TABLE `NEWS`.`comments` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `text` LONGTEXT NOT NULL,
  `pubtime` DATETIME NOT NULL,
  `new_id` INT(11) NOT NULL,
  `user_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_comments_new_news_id`
    FOREIGN KEY (`new_id`)
    REFERENCES `NEWS`.`news` (`id`),
  CONSTRAINT `fk_comments_user_id_auth_user_id`
    FOREIGN KEY (`user_id`)
    REFERENCES `NEWS`.`user` (`id`));


CREATE TABLE `NEWS`.`admin_log` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
```

```
  `action_time` DATETIME NOT NULL,
  `object_id` LONGTEXT NULL DEFAULT NULL,
  `object_repr` VARCHAR(200) NOT NULL,
  `action_flag` SMALLINT(5) UNSIGNED NOT NULL,
  `change_message` LONGTEXT NOT NULL,
  `content_type_id` INT(11) NULL DEFAULT NULL,
  `user_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_auth_user_id`,
  CONSTRAINT `fk_admin_log_user_auth_user_id`
    FOREIGN KEY (`user_id`)
    REFERENCES `NEWS`.`user` (`id`));


CREATE TABLE `NEWS`.`session` (
  `session_key` VARCHAR(40) NOT NULL,
  `session_data` LONGTEXT NOT NULL,
  `expire_date` DATETIME NOT NULL,
  PRIMARY KEY (`session_key`));


CREATE TABLE `NEWS`.`main_emailconfirm` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `token` VARCHAR(36) NOT NULL,
  `user_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `user_id` (`user_id` ASC) VISIBLE,
  CONSTRAINT `fk_main_emailconfirm_user__auth_user_id`
    FOREIGN KEY (`user_id`)
    REFERENCES `NEWS`.`user` (`id`));
```

ПРИЛОЖЕНИЕ В. Исходный код программного средства

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "news2.settings")

application = get_wsgi_application()
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'cna=lum&x)09x)-9n-gf#_v2!24ct$d@9f4pz*0hc&g2y#5dv7'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'main',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'news.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
```

```
            ],
        },
    },
]

WSGI_APPLICATION = 'news.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.0/topics/i18n/

LANGUAGE_CODE = 'ru-ru'

TIME_ZONE = 'Europe/Minsk'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC_URL = '/static/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'uploads')
MEDIA_URL = '/media/'

#!/usr/bin/env python3
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "news.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
from django.db import models
from main.models.news import News
from django.contrib.auth.models import User
```

```
class Coments(models.Model):

    class Meta:
        db_table = 'comments'
        verbose_name = 'коментарий'
        verbose_name_plural = 'коментарии'
        ordering = ['pubtime']

    new = models.ForeignKey(News, editable=False, on_delete=models.CASCADE)

    user = models.ForeignKey(
                             User,
                             verbose_name='Пользователь',
                             editable=False,
                             on_delete=models.SET_NULL,
                             blank=True,
                             null=True
                            )

    text = models.TextField(verbose_name='Коментарий')

    pubtime = models.DateTimeField(
                                   auto_now_add=True,
                                   editable=False,
                                   verbose_name='Дата коментирования'
                                   )

    def __str__(self):
        return  '{} к новости "{}" ({:%d.%m.%Y %H:%M})'\
                .format(self.user, self.new, self.pubtime)
from django.db import models
from django.core.validators import FileExtensionValidator
from django.conf import settings
from PIL import Image
import os
import uuid
import datetime


IMAGE_W = 640
IMAGE_H = 360

def image_name():
    cfg = {}
    cfg['root'] = 'news'
    cfg['now'] = datetime.datetime.now()
    cfg['rnd'] = uuid.uuid4()
    cfg['ext'] = 'jpg'

    return  '{root}/{now:%Y/%m/%d}/{rnd}.{ext}'.format(**cfg)

class News(models.Model):

    class Meta:
        db_table = 'news'
        verbose_name = 'Новость'
        verbose_name_plural = 'Новости'
        ordering = ['-pubtime']

    title = models.CharField(max_length=50, verbose_name='Заголовок')

    image = models.ImageField(
                              upload_to='temp',
                              verbose_name='Изображение'
                             )

    short_text = models.TextField(verbose_name='Краткое описание')

    text = models.TextField(verbose_name='Тескт')

    pubtime = models.DateTimeField(
                                   auto_now_add=True,
                                   editable=False,
                                   verbose_name='Дата публикации'
                                   )

    def __str__(self):
        return self.title
```

```
    def save(self, *args, **kwargs):
        if News.objects.filter(pk=self.pk).exists():
            try:
                this_record = News.objects.get(pk=self.pk)
                if this_record.image != self.image:
                    this_record.image.delete(save=False)
            except:
                print("WARNING: Can't delete file {}".format(thisself_record.image.path))

        super(News, self).save(*args, **kwargs)

        im = Image.open(self.image.path)
        im = im.resize((IMAGE_W, IMAGE_H))
        new_path = image_name()
        full_path = os.path.join(settings.MEDIA_ROOT, new_path)
        os.makedirs(os.path.split(full_path)[0], exist_ok=True)
        im.save(full_path)
        os.remove(self.image.path)

        self.image.name = new_path
        super(News, self).save(*args, **kwargs)
        print('ok')


    def delete(self, *args, **kwargs):
        try:
            this_record.image.delete(save=False)
        except:
            pass
        super(News, self).delete(*args, **kwargs)
from django.db import models
from django.contrib.auth.models import User
from uuid import uuid4

class EmailConfirm(models.Model):

    user = models.OneToOneField(User, on_delete=models.CASCADE)

    token = models.CharField(default=uuid4, max_length=36)
from main.models.news import News
from main.models.coments import Coments
from main.models.email_confirm import EmailConfirm
from django.contrib import admin

from main.models import News
from main.models import Coments

admin.site.register(News)
admin.site.register(Coments)
from django.views.generic.edit import CreateView
from main.models.news import News
from django.urls import reverse_lazy

class CreateNewView(CreateView):
    model = News
    fields = ['title', 'image', 'short_text', 'text']
    template_name = 'create.html'
    success_url = reverse_lazy('main')
from django.views.generic.base import View
from django.contrib.auth import logout
from django.shortcuts import redirect


class CustomLogoutView(View):

    def get(self, request, *args, **kwargs):
        logout(request)
        return redirect('main')
from django.views.generic.edit import FormView
from django.urls import reverse_lazy
from django.urls import reverse
from django.conf import settings
from django.contrib.auth.models import Group
from django.contrib.auth import logout

from main.forms import RegisterForm

from main.models import EmailConfirm
```

```
EMAIL_SUBJECT = 'Подтвердение почты'
EMAIL_TEXT = '{}'
DEFAULT_GROUP = 'User'

class CreateUserView(FormView):
    form_class = RegisterForm
    template_name = 'register.html'
    success_url = reverse_lazy('main')

    def form_valid(self, form, request):
        user = form.save(commit=False)
        user.is_active = False
        confirm = EmailConfirm()

        link = reverse('email.confirm', kwargs={'token' : confirm.token})
        url = request.build_absolute_uri(link)
        user.email_user(
                        EMAIL_SUBJECT,
                        EMAIL_TEXT.format(url),
                        from_email=settings.EMAIL_HOST_USER
                       )
        user.save()
        confirm.user = user
        group = Group.objects.get(name=DEFAULT_GROUP)
        user.groups.add(group)
        confirm.save()
        return super(CreateUserView, self).form_valid(form)

    def post(self, request, *args, **kwargs):
        form_class = self.get_form_class()
        form = self.get_form(form_class)
        if form.is_valid():
            return self.form_valid(form, request)
        else:
            return self.form_invalid(form)

    def get(self, request, *args, **kwargs):
        logout(request)
        return super(CreateUserView, self).get(self, request, *args, **kwargs)
from django.views.generic.list import ListView

from main.models import News

class MainView(ListView):
    template_name = 'main.html'
    model = News
    context_object_name = 'news'
from django.views.generic.edit import DeleteView
from main.models import Coments
from django.urls import reverse

class DeleteComentView(DeleteView):
    model = Coments
    template_name = 'coments_confirm_delete.html'
    context_object_name = 'coment'

    def get_success_url(self):
        pk = self.get_object().new.pk
        return reverse('detail', kwargs={'pk' : pk})
from django.views.generic.base import TemplateView

from main.models import EmailConfirm


class EmailConfirmView(TemplateView):

    template_name = 'email_confirm.html'

    def get_context_data(self, token, **kwargs):
        context = super().get_context_data(**kwargs)

        confirm = EmailConfirm.objects.filter(token=token)
        exst = confirm.exists()
        context['valid'] = exst

        if exst:
            confirm = confirm.get()
            user = confirm.user
```

```
            user.is_active = True
            confirm.delete()
            user.save()

        return context
from django.views.generic.list import ListView

from django.contrib.auth.models import User

class UsersView(ListView):
    template_name = 'users.html'
    model = User
    context_object_name = 'users'
from django.contrib.auth.views import LoginView
from django.contrib.auth import logout
from django.urls import reverse


class CustomLoginView(LoginView):

    template_name = 'login.html'

    def get(self, request, *args, **kwargs):
        logout(request)
        return super(CustomLoginView, self).get(self, request, *args, **kwargs)

    def get_success_url(self):
        return reverse('main')
from django.views.generic.edit import UpdateView
from main.models.news import News
from django.urls import reverse_lazy

class UpdateNewView(UpdateView):
    model = News
    fields = ['title', 'image', 'short_text', 'text']
    success_url = reverse_lazy('main')
    template_name = 'update.html'
from django.views.generic.detail import DetailView
from django.urls import reverse_lazy
from django.shortcuts import redirect
from datetime import timedelta
from datetime import datetime

from main.models import News
from main.models import Coments
from main.forms import ComentForm


COMMENT_TIME = timedelta(seconds=30)

class NewDetailView(DetailView):
    template_name = 'new.html'
    model = News
    context_object_name = 'new'
    login_url = reverse_lazy('login')

    form = None

    def get_context_data(self, **kwards):
        context = super(NewDetailView, self).get_context_data(**kwards)
        context['coment_form'] = self.form if self.form else ComentForm()
        context['coments'] = Coments.objects.filter(new=self.get_object())
        return context

    def post(self, request, *args, **kwargs):

        if not request.user.has_perm('main.add_coments'):
            return redirect(self.login_url)

        coment_form = ComentForm(request.POST)

        if not coment_form.is_valid():
            self.form = coment_form
            return super(NewDetailView, self).get(request, *args, **kwargs)

        coment = coment_form.save(commit=False)
        coment.user = request.user
        coment.new = self.get_object()
```

```
        if Coments.objects.filter(user=coment.user,
                                  pubtime__gte=datetime.now()-COMMENT_TIME
                                 ).exists():
            self.form = coment_form
            self.form.add_error('text',
                                "Нельзя отправлять комментарии чаще 30 секунд"
                                )
            return super(NewDetailView, self).get(request, *args, **kwargs)

        coment.save()

        return super(NewDetailView, self).get(request, *args, **kwargs)
from django.views.generic.edit import DeleteView
from main.models.news import News
from django.urls import reverse_lazy

class DeleteNewView(DeleteView):
    model = News
    success_url = reverse_lazy('main')
    template_name = 'news_confirm_delete.html'
    context_object_name = 'new'
from main.views.main import MainView
from main.views.detail import NewDetailView
from main.views.create import CreateNewView
from main.views.delete import DeleteNewView
from main.views.update import UpdateNewView
from main.views.create_user import CreateUserView
from main.views.delete_coment import DeleteComentView
from main.views.users import UsersView
from main.views.email_confirm import EmailConfirmView
from main.views.login import CustomLoginView
from main.views.logout import CustomLogoutView
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.password_validation import MinimumLengthValidator
from django.contrib.auth.password_validation import UserAttributeSimilarityValidator
from django.contrib.auth.password_validation import CommonPasswordValidator
from django.contrib.auth.password_validation import NumericPasswordValidator

from django.utils.translation import gettext as _

class RegisterForm(forms.ModelForm):

    error_messages = {
        'password_mismatch': _("The two password fields didn't match."),
        'email_occupation' : 'Данный адрес почты уже используется',
    }

    class Meta:
        model = User
        fields = ['username', 'first_name', 'last_name', 'email']


    def __init__(self, *args, **kwargs):
        super(RegisterForm, self).__init__(*args, **kwargs)
        for key in self.fields:
            self.fields[key].required = True

    password1 = forms.CharField(label=_("Password"),
            widget=forms.PasswordInput,
            validators=[
                        MinimumLengthValidator(min_length=8).validate,
                        UserAttributeSimilarityValidator().validate,
                        CommonPasswordValidator().validate,
                        NumericPasswordValidator().validate
                       ])

    password2 = forms.CharField(label=_("Password confirmation"),
        widget=forms.PasswordInput,
        help_text=_("Enter the same password as above, for verification.")
        )

    def clean_password2(self):
        password1 = self.cleaned_data.get("password1")
        password2 = self.cleaned_data.get("password2")
        if password1 and password2 and password1 != password2:
            raise forms.ValidationError(
                self.error_messages['password_mismatch'],
                code='password_mismatch',
```

```
            )
        return password2

    def clean_email(self, *args, **kwargs):
        email = self.cleaned_data.get("email")
        if User.objects.filter(email=email).exists():
            raise forms.ValidationError(
                self.error_messages['email_occupation'],
                code='email_occupation',
            )
        return email

    def save(self, commit=True):
        user = super(RegisterForm, self).save(commit=False)
        user.set_password(self.cleaned_data["password1"])
        if commit:
            user.save()

        return user
from django.forms import ModelForm
from main.models import Coments

class ComentForm(ModelForm):
    class Meta:
        model = Coments
        fields = ['text']
from main.forms.coment import ComentForm
from main.forms.register import RegisterForm
from django.conf.urls import url

from django.contrib.auth.decorators import permission_required
from django.urls import reverse_lazy

from main.views import MainView
from main.views import NewDetailView
from main.views import CreateNewView
from main.views import DeleteNewView
from main.views import UpdateNewView
from main.views import CreateUserView
from main.views import DeleteComentView
from main.views import UsersView
from main.views import EmailConfirmView
from main.views import CustomLoginView
from main.views import CustomLogoutView


login_url = reverse_lazy('login')


urlpatterns = [
    url(r'^$', MainView.as_view(), name='main'),

    url(r'^new/(?P<pk>\d+)/$', NewDetailView.as_view(), name='detail'),

    url(r'^new/add',
        permission_required('main.add_news', login_url=login_url)(CreateNewView.as_view()),
        name='new.add'),

    url(r'^new/(?P<pk>\d+)/delete/$',
        permission_required('main.delete_news', login_url=login_url)(DeleteNewView.as_view()),
name='new.delete'),

    url(r'^new/(?P<pk>\d+)/update/$',
        permission_required('main.change_news', login_url=login_url)(UpdateNewView.as_view()),
        name='new.update'),

    url(r'^login/$', CustomLoginView.as_view(), name='login'),

    url(r'^logout/$', CustomLogoutView.as_view(), name='logout'),

    url(r'^register/$', CreateUserView.as_view(), name='register'),

    url(
        r'^comments/delete/(?P<pk>\d+)$',
        permission_required('main.delete_coments',
login_url=login_url)(DeleteComentView.as_view()),
        name='coment.delete'
       ),
```

```
    url(r'^users/$', UsersView.as_view(), name='users'),

    url(r'^users/confirm/(?P<token>'
        r'[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}'
        r')$',
        EmailConfirmView.as_view(),
        name='email.confirm'),
]
from django.test import TestCase
```

ПРИЛОЖЕНИЕ Г. Разметка страниц программного средства

```
{% extends "base.html" %}

{% block title %} Delete::{{ new.title }} {% endblock %}


{% block content %}

<form action="" method="post">
    {% csrf_token %}
    <p>Подтверждаете удаление новости "{{ object }}"?</p>
    <input type="submit" value="Удалить" />
</form>

{% endblock %}
{% extends "base.html" %}

{% block title %} Delete::Comment {% endblock %}


{% block content %}

<p>Подтверждаете удаление коментария {{ coment }} :</p>
<p>{{ coment.text }} ?</p>

<form action="" method="post">
    {% csrf_token %}
    <input type="submit" value="Удалить" />
</form>

{% endblock %}
{% extends "base.html" %}

{% block title %}Регистрация{% endblock %}

{% block content %}

<form action="" method="post">
{% csrf_token %}
{{ form.as_p }}
<input type="submit" value="Регистрация">
</form>

{% endblock %}
{% extends "base.html" %}

{% block title %} Подтвердение почты {% endblock %}


{% block content %}

{%if valid %}
подтвержденно
{% else %}
неверный токен
{% endif %}

{% endblock %}
{% extends "base.html" %}

{% block title %}Вход{% endblock %}

{% block content %}

<form action="" method="post">
{% csrf_token %}
{{ form.as_p }}
<input type="submit" value="Вход">
</form>

{% endblock %}
{% extends "base.html" %}

{% block title %} {{ new.title }} {% endblock %}
```

```
{% block content %}

<form action="" method="post" enctype="multipart/form-data">

{{ form.as_p }}

{% csrf_token %}

<input type="submit" value="обновить">

</form>

{% endblock %}
{% extends "base.html" %}

{% block title %} Пользователи {% endblock %}

{% block content %}

{% for user in users %}
    <p>{{ user }}</p>
{% endfor %}


{% endblock %}

{% extends "base.html" %}

{% block title %} Главная {% endblock %}


{% block content %}

{% for new in news %}

<div class="new">
    <h2>{{ new.title }}</h2>
    <img src="{{ new.image.url }}">
    <div class="shorttext"> {{ new.short_text|linebreaks }} </div>
    <p class="time">{{ new.pubtime }}</p>
    <a href={% url 'detail' pk=new.pk %}>Подробнее</a>

    {% if perms.main.change_news %}
    <a href={% url 'new.update' pk=new.pk %}>Обновить</a>
    {% endif %}

    {% if perms.main.delete_news %}
    <a href={% url 'new.delete' pk=new.pk %}>Удалить</a>
    {% endif %}

</div>

{% endfor %}

{% if perms.main.add_news %}
<a href={% url 'new.add' %}>Добавить новость<a>
{% endif %}

{% endblock %}
<!doctype html>

<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>{% block title %}{% endblock %}</title>
        <link href="/static/css/styles.css" rel="stylesheet" media="all" />
    </head>
        <body>
            <div class="pageWrapper">

            <header>
            {% include "components/header.html" %}
            </header>

            <main>
            {% block content %}{% endblock %}
            </main>
```

```
            <footer>
            {% include "components/footer.html" %}
            </footer>

            </div>
        </body>
</html>
{% extends "base.html" %}

{% block title %} {{ new.title }} {% endblock %}


{% block content %}

<div class="details">

<h1>{{ new.title }}</h1>
<img src="{{ new.image.url }}">
<div class="text">{{ new.text|linebreaks }}</div>
<p class="time">{{ new.pubtime }}</p>

{% if perms.main.change_news %}
<a href={% url 'new.update' pk=new.pk %}>Обновить</a>
{% endif %}

{% if perms.main.delete_news %}
<a href={% url 'new.delete' pk=new.pk %}>Удалить</a>
{% endif %}
</div>


<div class="comments">
<h2>Коментарии</h2>
{% for comment in coments %}

<div class="comment">
<p class="name">{{ comment.user.get_username }}</p>
<p class="text">{{ comment.text }}</p>
<p class="time">{{ comment.pubtime }}</p>

{% if perms.main.delete_coments %}
<a href={% url 'coment.delete' pk=comment.pk %}>Удалить коментарий</a>
{% endif %}

</div>

{% endfor %}


{% if perms.main.add_coments %}
<form action="" method="post">
{% csrf_token %}
{{ coment_form.as_p }}
<input type="submit" value="Коментировать">
</form>
{% endif %}

</div>

{% endblock %}
<img src="/static/images/logo.png" class="logo" alt="logo">

<h1><a href="{% url "main" %}">Новости</a></h1>

<div class="login">
{% if not user.is_authenticated %}
<a href="{% url "login" %}">Войти</a>
<a href="{% url "register" %}">Регистрация</a>
{% else %}
<span>{{ user.username }}</span>
<a href="{% url "logout" %}">Выход</a>
{% endif %}

</div>
<div class="info">


<figure class="logo">
<img src="/static/images/logo.png">
```

```
<figcaption class="copy">
&copy; 2018 ЗАО "Крутая компания"
</figcaption>
</figure>

<small>
Свидетельство о регистрации №123456789 от 01.01.2000г.<br>
Лицензия на розничную торговлю №00000/00000 от 01.01.2000г.<br>
Дата включения в торговый реестр – 01.01.2000г.
</small>

<div class="contact">

<address>
emal: mega@cool.net<br>
MTC: +375 11 111 11 11<br>
VECOME: +375 22 222 22 22<br>
LIFE: +375 33 333 33 33
</address>

</div>

</div>
{% extends "base.html" %}

{% block title %} {{ new.title }} {% endblock %}


{% block content %}

<form action="" method="post" enctype="multipart/form-data">
{% csrf_token %}
{{ form.as_p }}
<input type="submit">
</form>

{% endblock %}
```

ПРИЛОЖЕНИЕ Д. Стили страниц программного средства

```
form
    font-size: 1.2em
    color: rgba(0, 0, 0, 0.3)

    p
        margin-bottom: 30px

    label
        display: inline-block
        width: 200px
        text-align: right
        margin-right: 10px
        font-weight: bold
        color: rgba(0,0,0,.5)

    input, textarea
        border-radius: 5px
        border: 1px solid rgba(0, 0, 0, 0.3)
        background-color: rgba(255,255,255,.5)
        color: rgba(0,0,0,0.6)
        font-size: 1.2em
        box-shadow: inset 0 0 10px rgba(255,255,255,.75)
        width: 300px
        padding: 5px

    input:focus, textarea:focus
        background-color: white
        color: black

    textarea
        width: 700px
        height: 200px
        margin: 0 30px
        margin-bottom: -20px

    .helptext
        display: block
        margin-left: 100px

    .errorlist
        color: red

    input[type="submit"]
        margin-left: 220px
        padding: 10px 25px
        font-size: 1em
        color: white
        background: rgba(0,102,153, 0.7)
        border-radius: 5px
        border: rgba(0,102,153, 0.6)
        width: 300px

        &:hover
            box-shadow: 0 0 3px 3px rgba(0,102,153, 0.2)

header
    display: flex
    color: white
    width: 100%
    justify-content: space-around
    align-items: center
    background: rgba(50, 50, 50, 0.9)
    position: fixed
    top: 0
    left: 0
    height: 70px

    .logo
        float: left
        box-sizing: border-box
        height: 80%


    h1
```

```
        text-transform: uppercase
        font-size: 3em

        a
            color: white
            text-decoration: none

    .login
        float: right
        margin: auto 0
        font-size: 1.5em

        span
            text-transform: uppercase
            margin-right: -10px

        a
            color: white
            border: 4px solid rgba(30,130,170, 1)
            border-radius: 15px
            padding: 5px 10px
            margin-left: 15px
            text-decoration: none

            &:hover
                background: rgba(30,130,170, 1)
body
    background: url("../images/background.png") repeat


.pageWrapper
    max-width: $page_heaght
    margin: 0 auto

    background: rgba(255, 255, 255, 0.8)
    border-radius: 50px
    box-shadow: 0 0 10px 5px rgba(30, 30, 30, 0.8)
footer

    background: rgba(0, 0, 0, 0.7)
    color: white
    border-radius: 0 0 50px 50px
    padding: 30px

    .info
        max-width: $page_heaght
        margin: 0 auto
        display: flex
        justify-content: space-between
        align-items: flex-start

    .logo
        height: 70px

    .logo img
        height: 100%
        margin-bottom: 5px

    .copy
        font-size: 0.8em

    small
        padding: 0 20px 0 20px
        margin-left: 30px

    .payments
        box-sizing: border-box
        clear: both
        display: block
        margin: 0 auto
        padding: 20px

main
    order: 1
    padding: 20px 50px
    margin-top: 100px


    .new
```

```
    border-radius: 20px
    padding: 0 80px 30px 80px
    margin: 50px 20px
    box-shadow: 0 0 10px
    background: rgba(200,200,200, 0.1)

    h2
        text-align: left
        font-size: 2.5em
        color: rgba(0, 0, 0, 0.7)
        margin-top: 10px
        text-shadow: 2px 2px 3px rgba(0,0,0,0.3)

    img
        margin: 30px auto
        border-radius: 20px
        display: block
        box-shadow: 0 0 30px

    .shorttext
        font-size: 1.5em
        margin-bottom: 20px

    .time
        float: right
        font-size: 1.5em
        color: rgba(30,130,170, 1)

    a
        font-size: 1.5em


.details

    padding: 0 50px

    h1
        font-size: 5em
        color: rgba(100, 100, 100, 0.9)
        text-align: center

    img
        margin: 50px auto
        border-radius: 20px
        display: block
        box-shadow: 0 0 30px

    .text
        font-size: 1.5em
        margin-bottom: 30px

    .time
        float: right
        font-size: 1.5em
        color: rgba(30,130,170, 1)

    a
        font-size: 1.5em
        margin-right: 50px


.comments

    padding: 0 50px
    margin-top: 50px

    h2
        font-size: 2em
        color: rgba(100, 100, 100, 0.9)
        text-align: left

    .comment

        margin: 40px 0
        border: 2px solid rgba(30,130,170, 0.8)
        border-radius: 10px
        box-shadow: 0 0 5px rgba(30,130,170, 0.8)

        .name
```

```
                font-size: 1.5em
                background: rgba(30,130,170, 0.8)
                border-radius: 5px 5px 0 0
                padding: 10px
                font-size: 1.5em

            .text
                font-size: 1.5em
                margin: 10px 0
                padding: 10px

            .time
                font-size: 1.3em
                color: rgba(30,130,170, 1)
                padding: 10px

            a
                font-size: 1.5em
                padding: 10px
                text-align: center
$page_heaght: 1000px
```