Министерство образования Республики Беларусь

Учреждение образования

«БелорусскиЙ государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Объектно-ориентированное программирование»

|  |
| --- |
| *К защите допустить*: |
| Руководитель курсовой работы  ассистент |
| \_\_\_\_\_\_\_\_\_\_\_\_А.В.Михалькевич  \_\_\_.\_\_\_. 20\_\_\_ |

**Пояснительная записка**

к курсовой работе

на тему

**Разработка базы данных и системы управления базой данных для геопортала**

БГУИР КР 1-40 05 01-10 №\_1

|  |  |  |
| --- | --- | --- |
| Студент | (подпись студента) | В.В. Ананенко |
|  |  | Курсовая работа представлена на проверку \_\_\_.\_\_\_. 20\_\_\_ |
|  |  | (подпись студента) |

Минск 2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc6294113)

[1 ОПИСАНИЕ ПРОЕКТА 6](#_Toc6294114)

[2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ 7](#_Toc6294115)

[3 ИНСТРУМЕНТАРИЙ 9](#_Toc6294116)

[3.1 Обоснование используемых инструментов 9](#_Toc6294117)

[3.2 Использование системы контроля версий GIT 10](#_Toc6294118)

[4 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ MVC 14](#_Toc6294119)

[5 РАЗРАБОТКА БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ 17](#_Toc6294120)

[6 ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 19](#_Toc6294121)

[ЗАКЛЮЧЕНИЕ 24](#_Toc6294122)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc6294123)

[ПРИЛОЖЕНИЕ А (обязательное) Листинг кода 26](#_Toc6294124)

**ВВЕДЕНИЕ**

В настоящее время Интернет становиться все более развитой средой для осуществления коммуникаций с потребителями, а также инструментом для упрощения их жизней. В то же время, существенным является и тот факт, что Интернет становиться удобным и достаточно доступным сервисом для поиска нужной информации.

Все большее количество разработчиков старается создать удобные и легкие в использовании программы, сайты, порталы для людей, которые интересуются новостями Мира или же хотят здесь и сейчас иметь доступ к информации об определенной стране, регионе, городе.

При этом, с помощью интернет-порталов можно с легкостью находить информацию, представленную в доступном визуале, который сможет понять любой пользователь.

Определенную часть аудитории сети Интернет составляют люди, проводящие большую часть времени в поездках, командировках, путешествиях. Как правило, их образ жизни не позволяет им тратить большое количество времени на сбор информации из различных online- или offline-источников о местах, где им предстоит провести время. Возможность быстро найти нужную и свежую информацию об интересующей их стране в Интернете является для них действительно выходом из ситуации.

В соответствии с терминологией в области геоинформатики и инфраструктуры пространственных данных (ИПД) геопортал можно определить как «точку входа в Интернет или Интранет с инструментами просмотра метаданных, поиска географической информации, ее визуализации, загрузки, распространения и, возможно, поиска геосервисов». Кроме традиционных функций по визуализации карт, они предоставляют значения координат интересующих объектов с возможностью подсчета их периметра или площади, прокладки маршрутов и печати.

По совокупности важных отличительных признаков геопорталов особо можно выделить следующие:

* доступ к картографическим данным из различных источников;
* визуализация и поиск данных на карте;
* прикладные функциональные возможности.

В курсовой работе поставлена цель: проектирование геопортала.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучение современных инструментов разработки сайтов.
2. Разработка макета геопортала.
3. Создание баз данных для хранения информации.
4. Интеграция сервиса для работы с картами.
5. Организация автоматической загрузки информации, запрашиваемой пользователем.
6. **ОПИСАНИЕ ПРОЕКТА**

Курсовой проект представляет собой географический веб-портал с возможностью авторизации при помощи электронной почты, загрузки последних новостей для каждой страны, сохранения предыдущих запросов новостей для быстрого доступа к уже имеющейся информации. Фото и видео контент также автоматически загружается при обновлении страницы, обеспечивая актуальность запрашиваемой пользователем информации. Для каждой статьи отображается ссылка для перехода к источнику информации и возможности более детального ознакомления с ней. Геопортал обеспечивает пользователем уникальную возможность доступа к разнообразной информации, касающейся каждой страны.

На главной странице портала представлена интерактивная карта Мира с территориальным разделением по странам, что обеспечивает простую навигацию по сайту и простоту использования. В свою очередь, страны подразделяются на регионы и города, для каждой страны имеется полный их список. При переходе к выбранным городам пользователь имеет возможность посмотреть на карте точное местоположение данного объекта.  
При нахождении пользователя на главное странице, на карте отображается его текущее местоположение, что очень удобно.

В каждом разделе имеется подробная и последняя информация в виде новостей и видео о месте, которая автоматически загружается при запросе пользователя.

Геопортал разработан при помощи языка PHP, с использованием языков разметки HTML и CSS, а также языка сценариев JavaScript[1], [2].

Вся информация, которая отображается на сайте, хранится при помощи баз данных, доступ к которым осуществляется по запросам.

Сайт реализован на нескольких языках: русский, английский, немецкий, французский, итальянский. Пользователь может сделать выбор на любом этапе работы с сайтом, не переходя на главную страницу.

Навигация по сайту довольно проста и интуитивно понятна, использование интерактивных карт и анимаций информирует пользователя о загрузке запрашиваемых данных.

1. **ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ**

Событие в объектно-ориентированном программировании – это сообщение, которое возникает в различных точках исполняемого кода при выполнении определённых условий. События предназначены для того, чтобы иметь возможность предусмотреть реакцию программного обеспечения. Для решения поставленной задачи создаются обработчики событий: как только программа попадает в заданное состояние, происходит событие, посылается сообщение, а обработчик перехватывает это сообщение. В общем случае в обработчик не передаётся ничего, либо передаётся ссылка на объект, инициировавший (породивший) обрабатываемое событие. В особых случаях в обработчик передаются значения некоторых переменных или ссылки на какие-то другие объекты, чтобы обработка данного события могла учесть контекст возникновения события. Самое простое событие – это событие, сообщающее о начале или о завершении некоторой процедуры. Событие, по сути, сообщает об изменении состояния некоторого объекта. Наиболее наглядно события представлены в пользовательском интерфейсе, когда каждое действие пользователя порождает цепочку событий, которые, затем обрабатываются в приложении. В объектно-ориентированном анализе для описания динамического поведения объектов принято использовать модель состояний. Событие - это переход объекта из одного состояния в другое. Взаимодействие объектов также осуществляется при помощи событий: изменение состояния одного объекта приводит к изменению состояния другого объекта, а событие оказывается средством связи между объектами. Событие – это абстракция инцидента или сигнала в реальном мире, который сообщает нам о перемещении чего- либо в новое состояние [3].

В объектно-событийном программировании выполнение программы определяется событиями – действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета).

Объектно-событийное программирование можно также определить как способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события. Развитием объектно-ориентированной парадигмы (методологии), стала объектно-событийная парадигма, опирающаяся на понятия объекта и события. Эта парадигма позволяет конструировать, программировать распределенные вычислительные среды, в том числе среды реального времени, SCADA и пр.

Выделяются четыре аспекта события:

* метка – уникальный идентификатор события.
* значение – текстовое сообщение о сути произошедшего.
* предназначение – модель событий, которая принимает событие.
* данные – данные, которые переносятся от одного объекта к другому.

Первый ряд примеров событий доставляет собственно сам жизненный цикл объекта:

* создание объекта;
* уничтожение объекта.

Более сложные примеры событий возникают тогда, когда у объекта появляются внутренние состояния, которые описываются соответствующей диаграммой переходов (из одного состояния в другое).

# 3 ИНСТРУМЕНТАРИЙ

* 1. **Обоснование используемых инструментов**

Open Server – серверная платформа, разработанная для нужд и с учетом рекомендаций веб-разработчиков. В отличие от аналогичных пакетов, не нуждается в установке, может использоваться с портативного накопителя и оснащена большим количеством дополнительных программ.

Open Server позволяет разрабатывать, выполнять отладку, тестировать веб-проекты, представлять веб-сервисы в локальных сетях. Программы, которыми укомплектована платформа, помогают пользователю в работе с графикой, редактировании кода, текста, архивировании, записи дисков, работе в сети интернет.

Основные возможности Open Server:

* быстрый запуск и завершение работы;
* отсутствие привязки к конкретному ПК;
* автозапуск сервера при запуске управляющего ПО;
* управление доменами в нескольких режимах;
* возможность монтирования виртуального диска;
* управление через командную строку;
* демонстрация логов компонентов;
* функция переключения между модулями HTTP, MySQL, PHP;
* работа комплекса программ на локальном, сетевом и внешнем IP адресе;
* поддержка SSL без дополнительных настроек;
* создание домена при помощи создания простой папки;
* конвертирование доменных имен;
* поддержка доменов на кириллице, доменных указателей;
* обеспечение защиты сервера от доступа извне;
* возможность создания локального поддомена и обеспечение одновременной видимости основного домена в сети.

Notepad++ - текстовый редактор, предназначенный для программистов и всех тех, кого не устраивает скромная функциональность входящего в состав Windows Блокнота.

Основные особенности программы:

* подсветка текста и возможность сворачивания блоков, согласно синтаксису языка программирования;
* поддержка большого количества языков (C, C++, Java, XML, HTML, PHP, Java Script, ASCII, VB/VBS, SQL, CSS, Pascal, Perl, Python, Lua, TCL, Assembler);
* WYSIWYG (печатаешь и получаешь то, что видишь на экране);
* настраиваемый пользователем режим подсветки синтаксиса;
* авто-завершение набираемого слова;
* одновременная работа с множеством документов;
* одновременный просмотр нескольких документов;
* поддержка регулярных выражений Поиска/Замены;
* полная поддержка перетягивания фрагментов текста;
* динамическое изменение окон просмотра;
* автоматическое определение состояния файла;
* увеличение и уменьшение;
* заметки;
* выделение скобок при редактировании текста;
* запись макроса и его выполнение.

OpenStreetMap – проект по созданию и распространению свободных географических данных, например, дорожных карт.

С помощью OpenStreetMap вы можете сами дорисовать недостающую или исправить существующую часть карты.

В настоящее время очень хорошо прорисованы некоторые страны Европы (Голландия, Германия, Англия - вплоть до проселочных дорог) и США. Активно рисуется Индия.

Касательно содержимого карт, OpenStreetMap является и нейтральной и прозрачной. OpenStreetMap является подобной Вики картой, которую может редактировать кто угодно. Если магазин отсутствует на карте, его может добавить как владелец магазина, так и его посетитель. Что касается показа (рендеринга) карты, то любой человек или компания, принимающие участие в создании карты, свободен рендерить её как ему удобно. Главная карта на OpenStreetMap.org использует ПО рендеринга и стиль со свободной лицензией, которые кто угодно может взять и подправить под свои нужды. Проще говоря, любой, кому необходимо, всегда может создать свои собственные карты, основываясь на данных OSM.

Также, несмотря на то, что самые популярные построители маршрутов для OpenStreetMap лицензированы под FLOSS, даже если какая-нибудь компания и выберет другую лицензию, пользователи всегда могут использовать свои построители маршрутов, и сравнив результаты построения, выявить какие-либо подтасовки, если они есть.

И наконец, пользователь волен скачать любую часть или даже всю карту OpenStreetMap для использования в оффлайне. Это означает, что можно использовать данные OpenStreetMap для навигации, вообще не передавая информацию о вашем местоположении на сторону.

OpenStreetMap уважает сообщества и уважает людей.

* 1. **Использование системы контроля версий GIT**

Система контроля версий (СКВ) – это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов [4].

СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, испортить или потерять файлы, всё можно будет легко восстановить.

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (рисунок 1).



Рисунок 1- Схема локальной СКВ

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч – файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (рис. 2).



Рисунок 2 - Схема централизованного контроля версий

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный – централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё – всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (рисунок 3).



Рисунок 3 – Схема распределённой системы контроля версий

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

В нашем курсовом проекте использовался онлайн-репозиторий GitHub. Ссылка на него: <https://github.com/VeronikaAnanenko/Geography>.

**4 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ MVC**

Шаблон проектирования Модель – Представление – Контроллер  
(MVC) – это шаблон программной архитектуры, построенный на основе сохранения представления данных отдельно от методов, которые взаимодействуют с данными [5].

Не смотря на то, что схема MVC была первоначально разработана для персональных компьютеров, она была адаптирована и широко используется веб-разработчиками из-за точного разграничения задач и возможности повторного использования кода. Схема стимулирует развитие модульных систем, что позволяет разработчикам быстро обновлять, добавлять или удалять функционал.

Название шаблона проектирования определяется тремя его основными составляющими частями: Модель, Представление и Контроллер. Визуальное представление шаблона MVCвыглядит, как показано на [приведенной ниже диаграмме](http://en.wikipedia.org/wiki/File:MVC-Process.png) (рис.4):

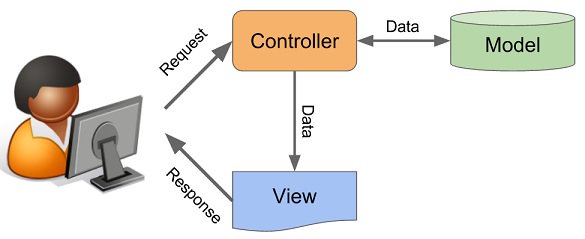


Рисунок 4 - Схема шаблона MVC

На рисунке показана структура одностороннего потока данных и пути его следования между различными компонентами, а также их взаимодействие.

Моделью называют постоянное хранилище данных, используемых во всей структуре. Она должна обеспечивать доступ к данным для их просмотра, отбора или записи. В общей структуре Модель является мостом между компонентами Представление и Контроллер.

При этом Модель не имеет никакой связи или информации о том, что происходит с данными, когда они передаются компонентам Представление или Контроллер. Единственная задача Модели – обработка данных в постоянном хранилище, поиск и подготовка данных, передаваемых другим составляющим MVC.

Модель должна выступать в качестве «привратника», стоящего возле хранилища данных и не задающего вопросов, но принимающего все поступающие запросы. Зачастую это наиболее сложная часть системы MVC. Компонент Модель – это вершина всей структуры, так как без нее невозможна связь между Контроллером и Представлением.

Представление – это часть системы, в которой данным, запрашиваемым у Модели, задается окончательный вид их вывода. В веб-приложениях, созданных на основе MVC, Представление – это компонент, в котором генерируется и отображается HTML-код.

Представление также перехватывает действие пользователя, которое затем передается Контроллеру. Характерным примером этого является кнопка, генерируемая Представлением. Когда пользователь нажимает ее, запускается действие в Контроллере.

Существует несколько распространенных заблуждений относительно компонента Представление. Например, многие ошибочно полагают, что Представление не имеет никакой связи с Моделью, а все отображаемые данные передаются от Контроллера. В действительности такая схема потока данных не учитывает теорию, лежащую в основе MVCархитектуры.

Кроме этого определение Представления как файла шаблона также является неточным. Но это не вина одного человека, а результат множества ошибок различных разработчиков, которые приводят общему заблуждению. После чего они неправильно объясняют это другим. На самом деле Представление это намного больше, чем просто шаблон. Но современные MVC-ориентированные фреймворки до такой степени впитали этот подход, что никто уже не заботится о том, поддерживается ли верная структура MVC или нет.

Компоненту Представление никогда не передаются данные непосредственно Контроллером. Между Представлением и Контроллером нет прямой связи – они соединяются с помощью Модели.

Его задача заключается в обработке данных, которые пользователь вводит и обновлении Модели. Это единственная часть схемы, для которой необходимо взаимодействие пользователя.

Контроллер можно определить, как сборщик информации, которая затем передается в Модель с последующей организацией для хранения. Он не содержит никакой другой логики, кроме необходимости собрать входящие данные. Контроллер также подключается только к одному Представлению и одной Модели. Это создает систему с односторонним потоком данных с одним входом и одним выходом в точках обмена данными.

Контроллер получает задачи на выполнение только когда пользователь взаимодействует с Представлением, и каждая функция зависит от взаимодействия пользователя с Представлением. Наиболее распространенная ошибка разработчиков заключается в том, что они путают Контроллер со шлюзом, поэтому присваивают ему функции и задачи, которые относятся к Представлению.

Также распространенной ошибкой является наделение Контроллера функциями, которые отвечают только за обработку и передачу данных из Модели в Представление. Но согласно структуре MVC паттерна это взаимодействие должно осуществляться между Моделью и Представлением.

1. **РАЗРАБОТКА БАЗЫ ДАННЫХ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ**

Каждый владелец сайта знает, что для правильного функционирования сайта нужны не только файлы с кодом страниц, но и базы данных. Для взаимодействия с базами данных используются системы управления базами данных (СУБД).

База данных представляет собой определенный набор данных, которые, как правило, связаны объединяющим признаком либо свойством (или несколькими). Эти данные упорядочены, например, по алфавиту. Обилие различных данных, которые могут быть помещены в единую базу, ведет к множеству вариаций того, что может быть записано: личные данные пользователей, записи, даты, заказы и так далее.

В первую очередь это удобно тем, что информацию можно быстро заносить в базу данных и так же быстро ее извлекать при необходимости. Если на заре развития web-разработки все необходимые данные нужно было прописывать в коде страницы, то теперь такая необходимость отсутствует – нужная информация может быть запрошена из базы данных при помощи скриптов. Специальные алгоритмы хранения и поиска информации, которые используются в базах данных, позволяют находить нужные сведения буквально за доли секунд – а при работе в виртуальном пространстве скорость работы ресурса важна как ничто другое.

Немаловажной является и взаимосвязь информации в базе данных: изменение одной строчки может привести к значительным изменениям других строк. Работать с данными таким образом гораздо проще и быстрее, чем, если бы изменения касались только одного места в базе данных.

Как можно догадаться уже из названия, система управления базами данных (или сокращенно СУБД) представляет собой программное обеспечение, которое используется для создания и работы с базами данных. Главная функция СУБД – это управление данными (которые могут быть как во внешней, так и в оперативной памяти). СУБД обязательно поддерживает языки баз данных, а также отвечает за копирование и восстановление данных после каких-либо сбоев.

Что касается классификации баз данных, то тут возможны различные варианты. К примеру, можно разделить базы по модели данных: иерархические (имеют древовидную структуру), сетевые (по своей структуре похожи на иерархические), реляционные (используются для управления реляционными базами данных), объектно-ориентированные (используются для объектной модели данных) и объектно-реляционные (некое слияние реляционного и объектно-ориентированного вида баз данных).

Либо, если деление идет по тому, где размещается СУБД, их можно разделить на локальные – вся СУБД размещается на одном компьютере, и распределенные – части системы управления базами данных находятся на нескольких компьютерах.

Файл-серверные, клиент-серверные и встраиваемые – такие названия носят СУБД, если разделить их по способу доступа к базам данных. Файл-серверные СУБД на данный момент уже считаются устаревшими; в основном идет использование клиент-серверных (СУБД, которые располагаются на сервере вместе с самой базой данных) и встраиваемых (не требующих отдельной установки) систем.

Информация, которая хранится в базах данных, не ограничивается только текстовыми или графическими файлами – современные версии СУБД поддерживают также форматы аудио и видеофайлов.

Зачем же нужны эти СУБД? Помимо основной своей функции – хранения и систематизации огромного количества информации – они позволяют быстро обрабатывать клиентские запросы и выдавать свежую и актуальную информацию.

Реляционные и объектно-реляционные СУБД являются одними из самых распространенных систем. Они представляют собой таблицы, у которых каждый столбец (который называется “field” или «поле») упорядочен и имеет определенное уникальное название. Последовательность строк (их называют “records” или «записи») определяется последовательностью ввода информации в таблицу. При этом обрабатывание столбцов и строк может происходить в любом порядке. Таблицы с данными связаны между собой специальными отношениями, благодаря чему с данными из разных таблиц можно работать – к примеру, объединять их – при помощи одного запроса.

Для управления реляционными базами данных применяется особый язык программирования – SQL. Сокращение расшифровывается как “Structured query language”, в переводе на русский «язык структурированных запросов».

Команды, которые используются в SQL, делятся на те, которые манипулируют данными, те, которые определяют данные, и те, которые управляют данными.

Схема работы с базой данных представлена на рисунке 5:

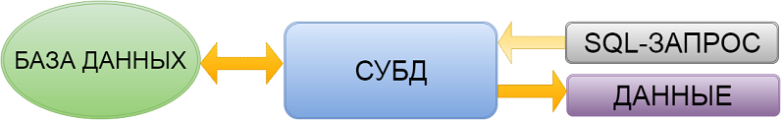


Рисунок 5 - Схема работы с базой данных

1. **ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА**

При загрузке геопортала, пользователь попадает на главную страницу сайта, где отображена интерактивная карта мира.

Сайт реализован на нескольких языках: русский, английский, немецкий, французский, итальянский. Выбор языка осуществляется в правом верхнем углу (рис.6).

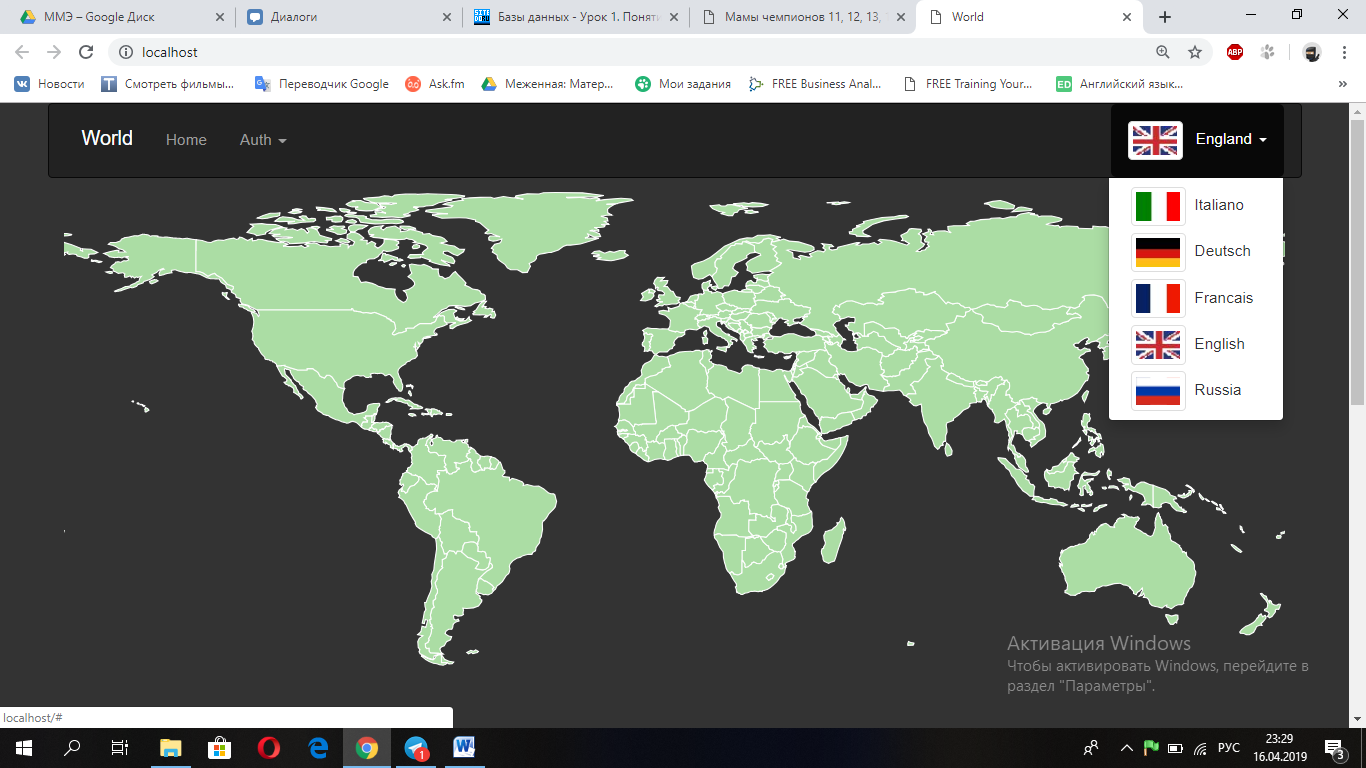


Рисунок 6 – Выбор языка

Чтобы выбрать страну, достаточно навести курсор. Выбранная страна выделится красным цветом. Пример показан на рисунке 7:

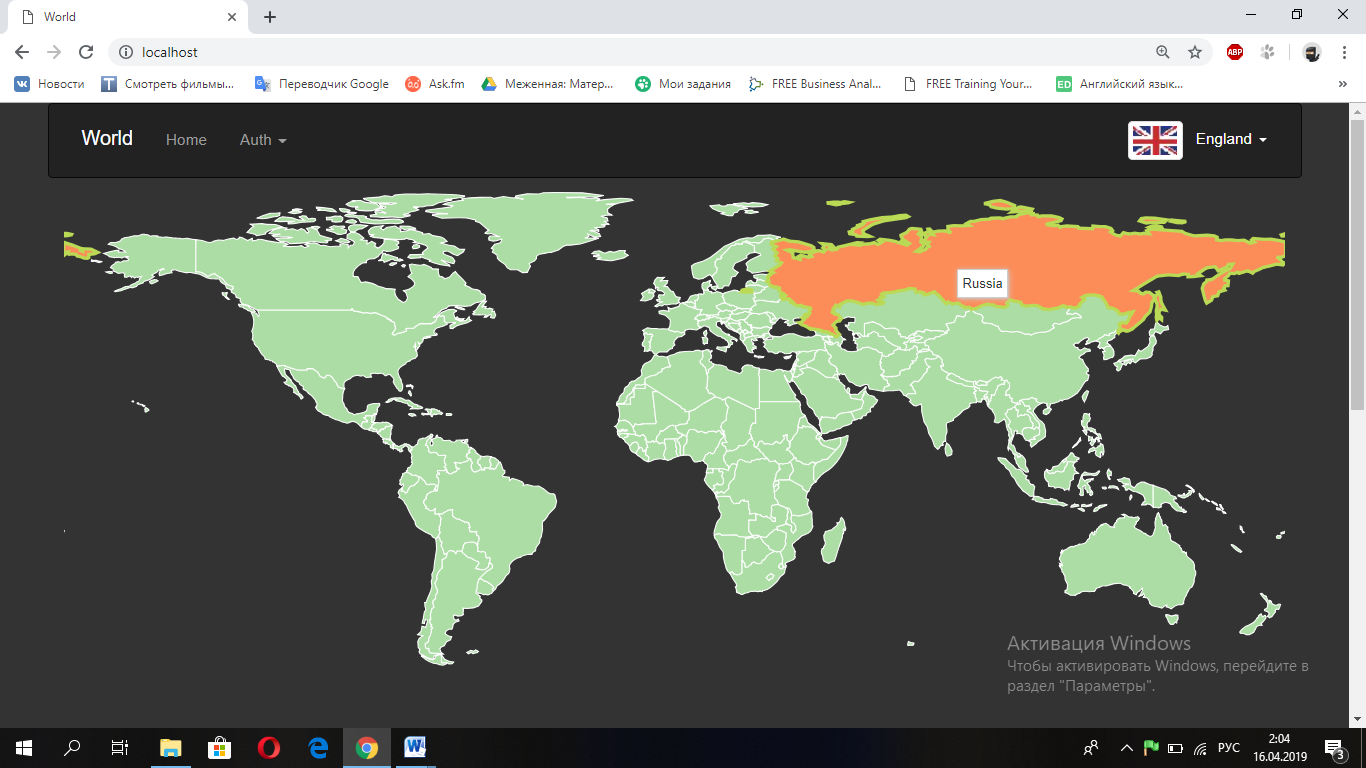


Рисунок 7 – Выбор страны на главной странице

Под картой отображаются новости, загруженные пользователем ранее.

При переходе на выбранную страну, появляется поле с информацией о ней. В нем содержится таблица с основными данными, а так же дополнительная информация в виде текста с картинками (рис.8).

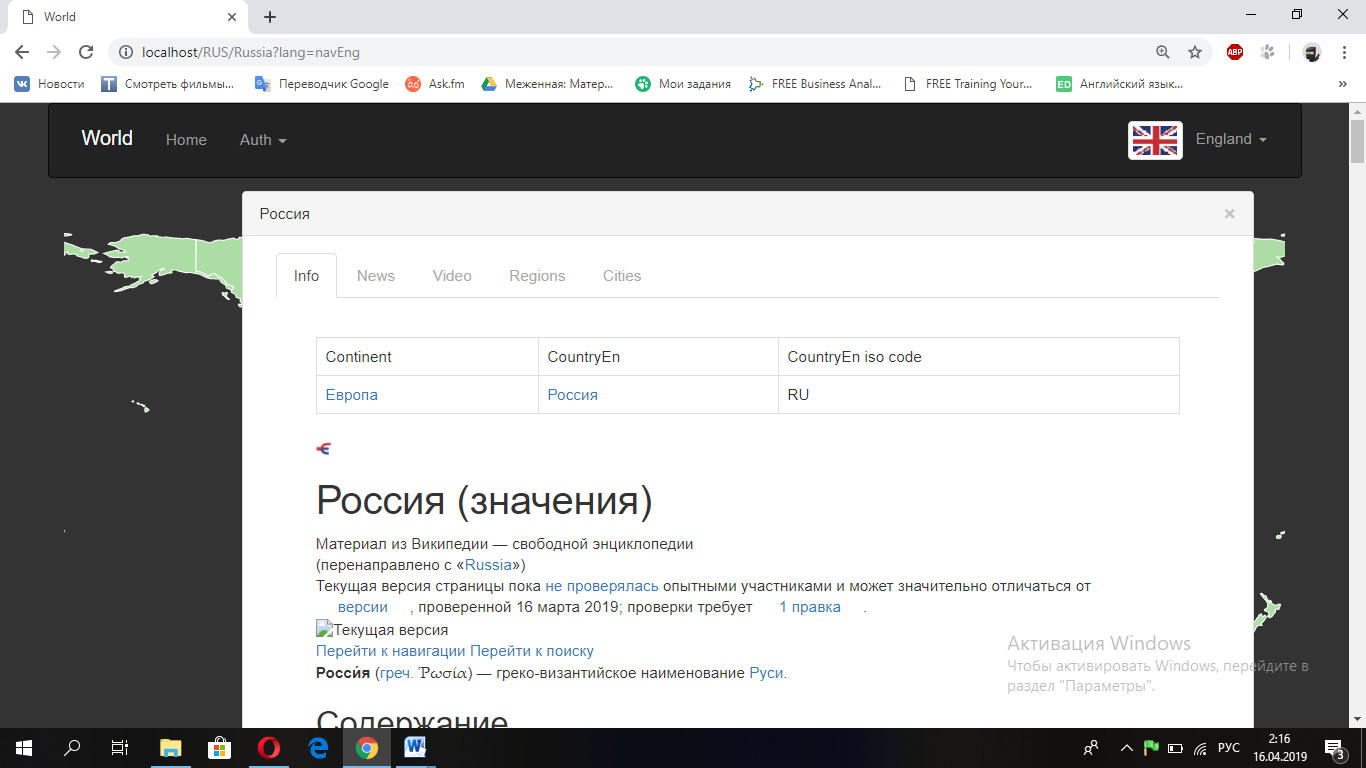


Рисунок 8 – Вкладка с информацией

Также, в горизонтальном меню можно перейти в графу «Новости», где по запросу отображаются актуальные новости выбранной страны, а так же ссылки на полную статью с данной информацией (рис.9).

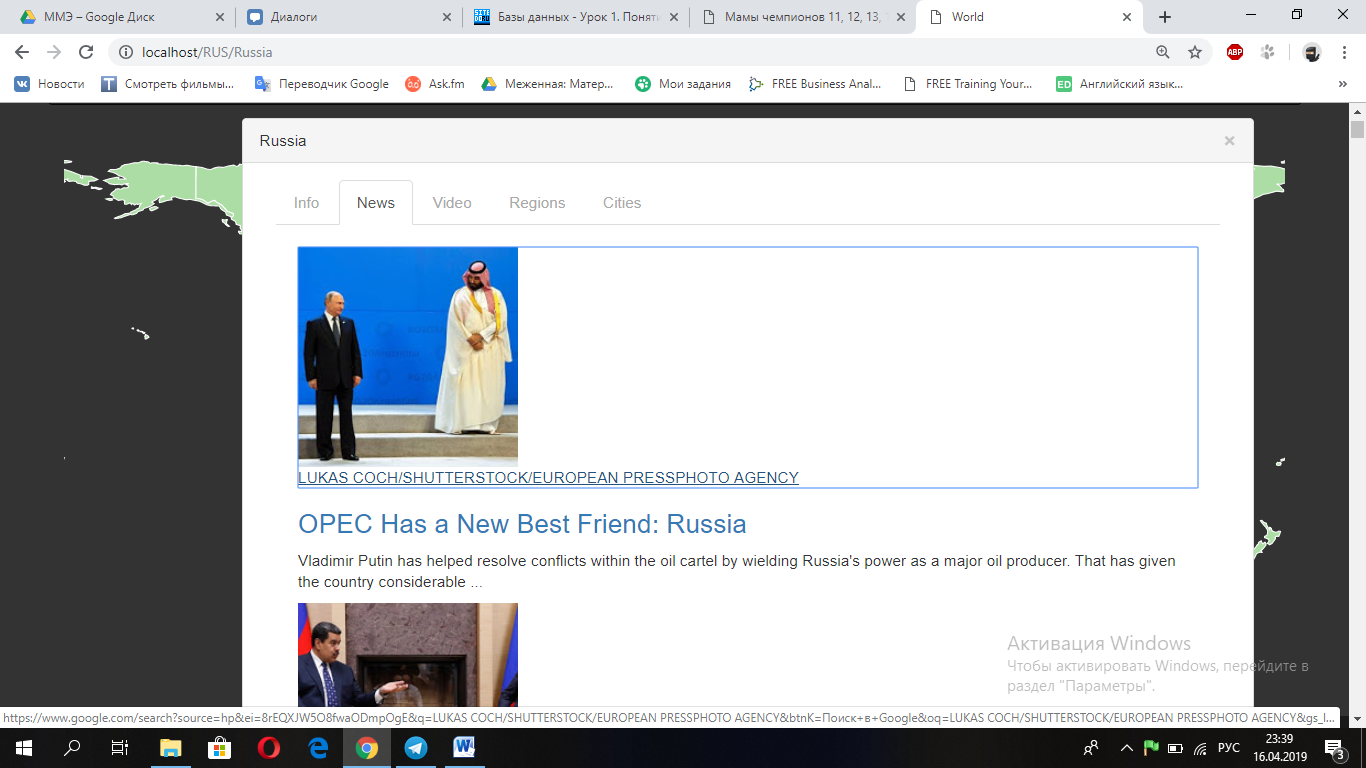


Рисунок 9 – Вкладка новостей

Следующим разделом информации, связанной с выбранной страной, является раздел «Видео» (рис.10). Здесь отображаются видеоролики с указанием оригинальной ссылки на них и длительности самого видеоролика.

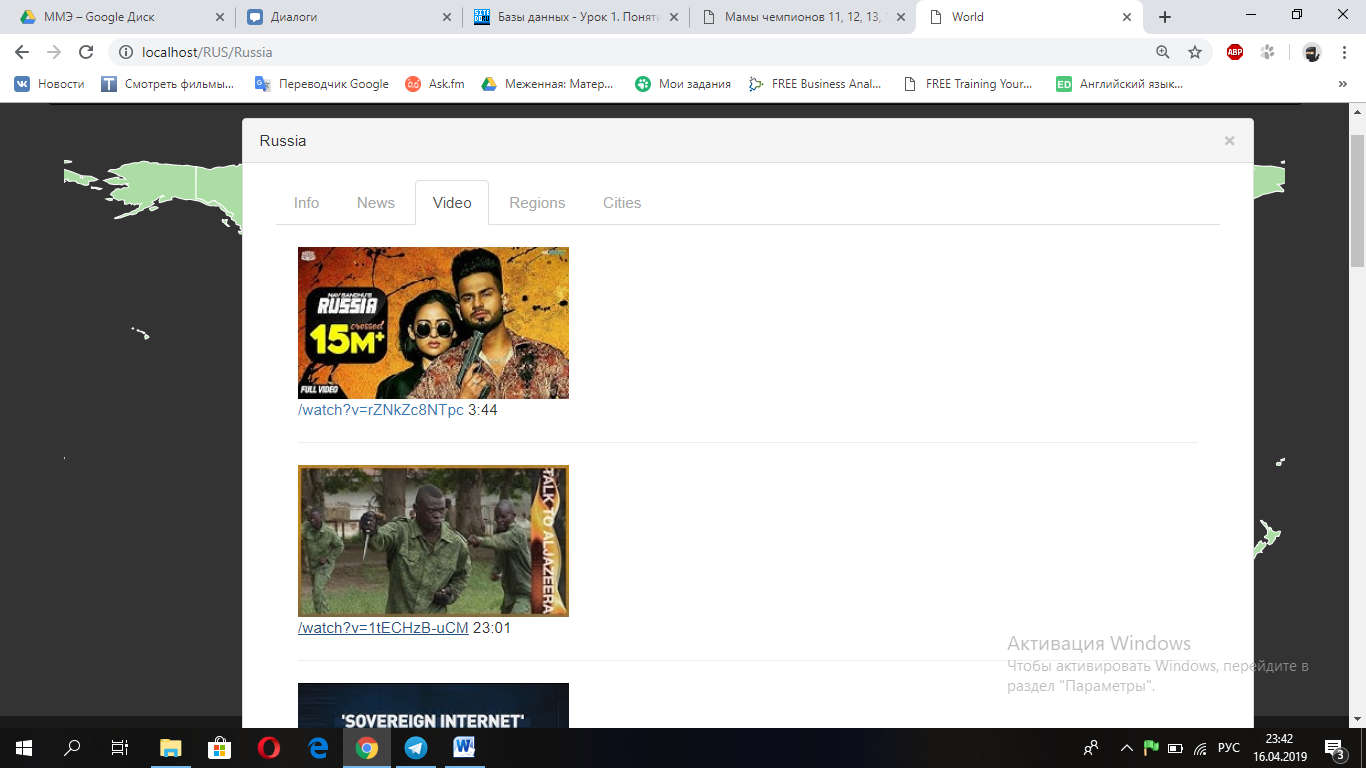


Рисунок 10 – Вкладка с видео

На вкладках «Регионы» и «Города» отображается соответственно список регионов и городов в алфавитном порядке. Пример представлен на рисунке 11:

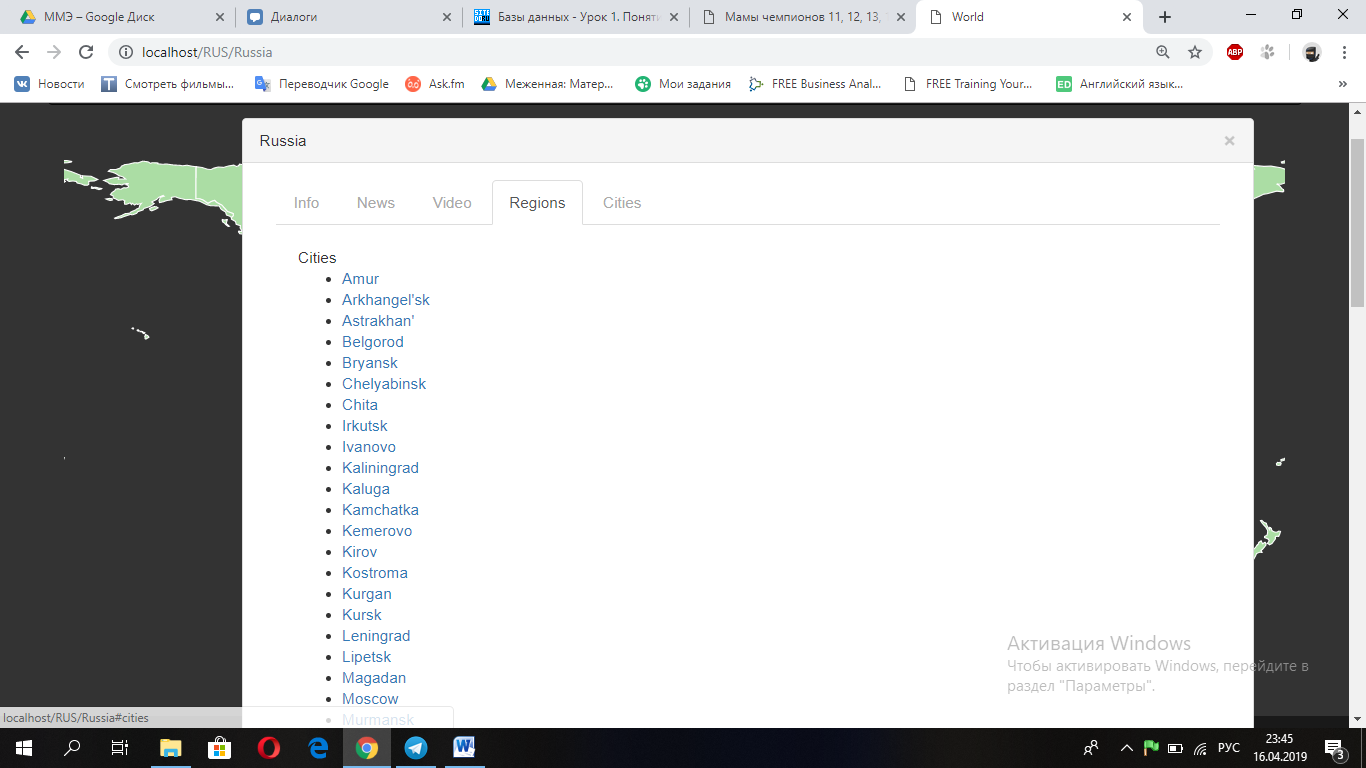


Рисунок 11 – Вкладка «Регионы»

При выборе конкретного города отображается карта страны с указанием местоположения данного места, как на рисунке 12.

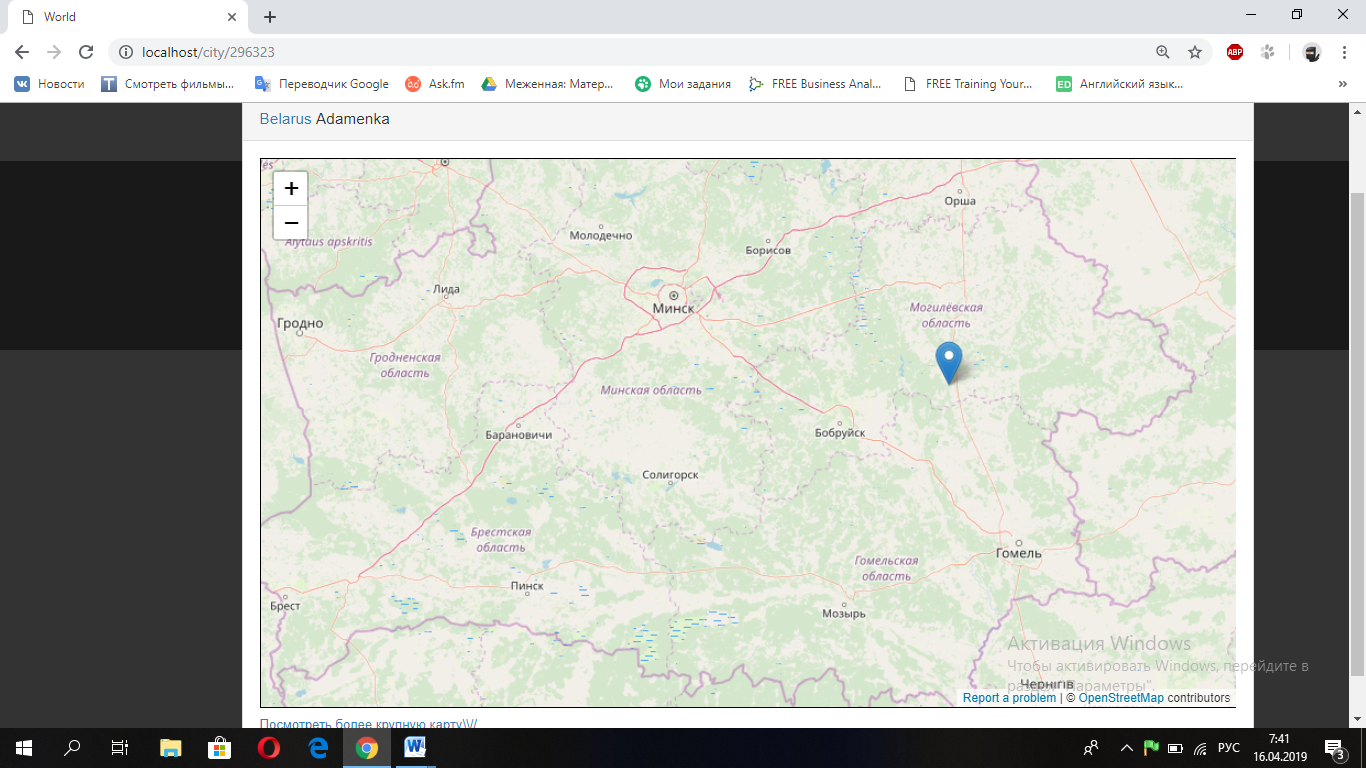


Рисунок 12 – Окно с выбранным городом

На странице «Дом» отображается карта страны с уже определенным местоположением пользователя. Например, на рисунке 13 отображена карта Беларуси.

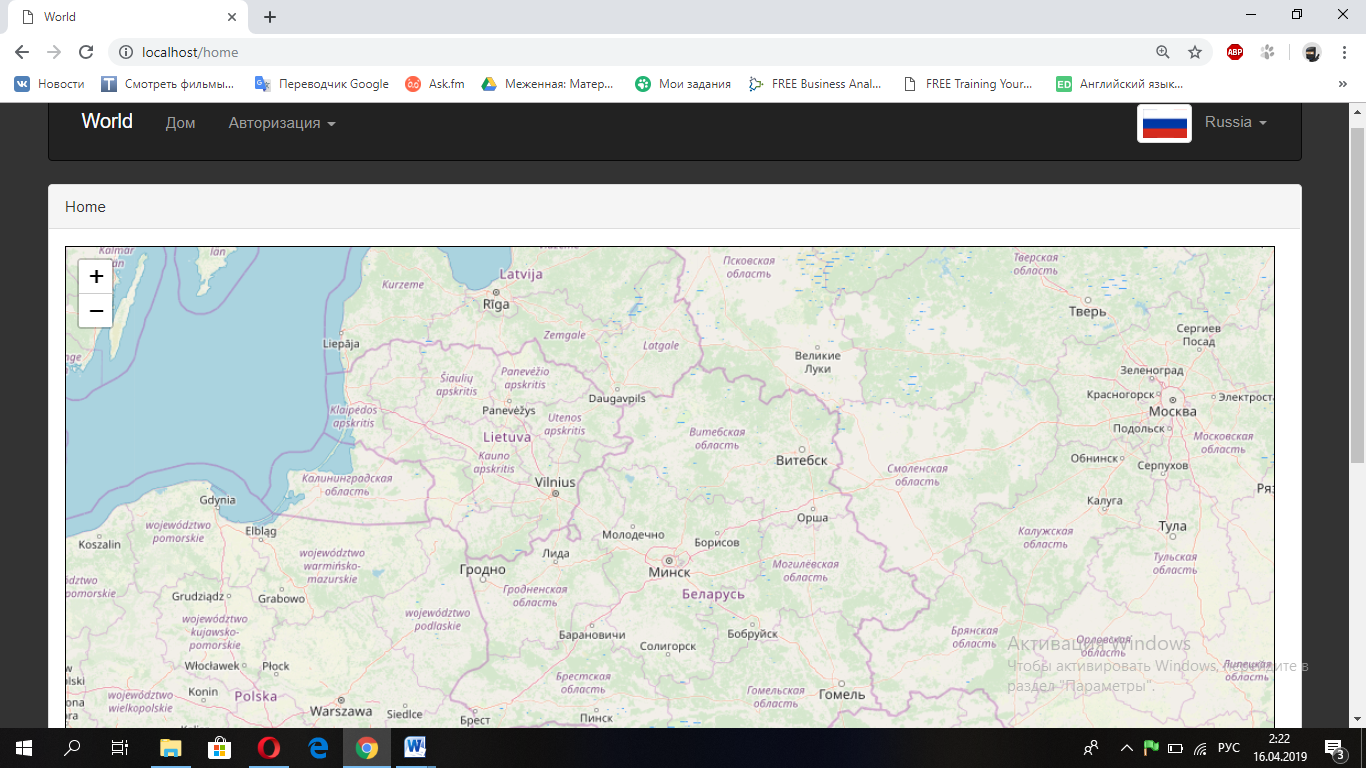


Рисунок 13 – Домашняя страница

В верхнем левом углу экрана можно выбрать два действия, связанных с аккаунтом пользователя: «Регистрация» и «Логин через e-mail».

При выборе первого варианта отображается окно регистрации, представленное на рисунке 14.

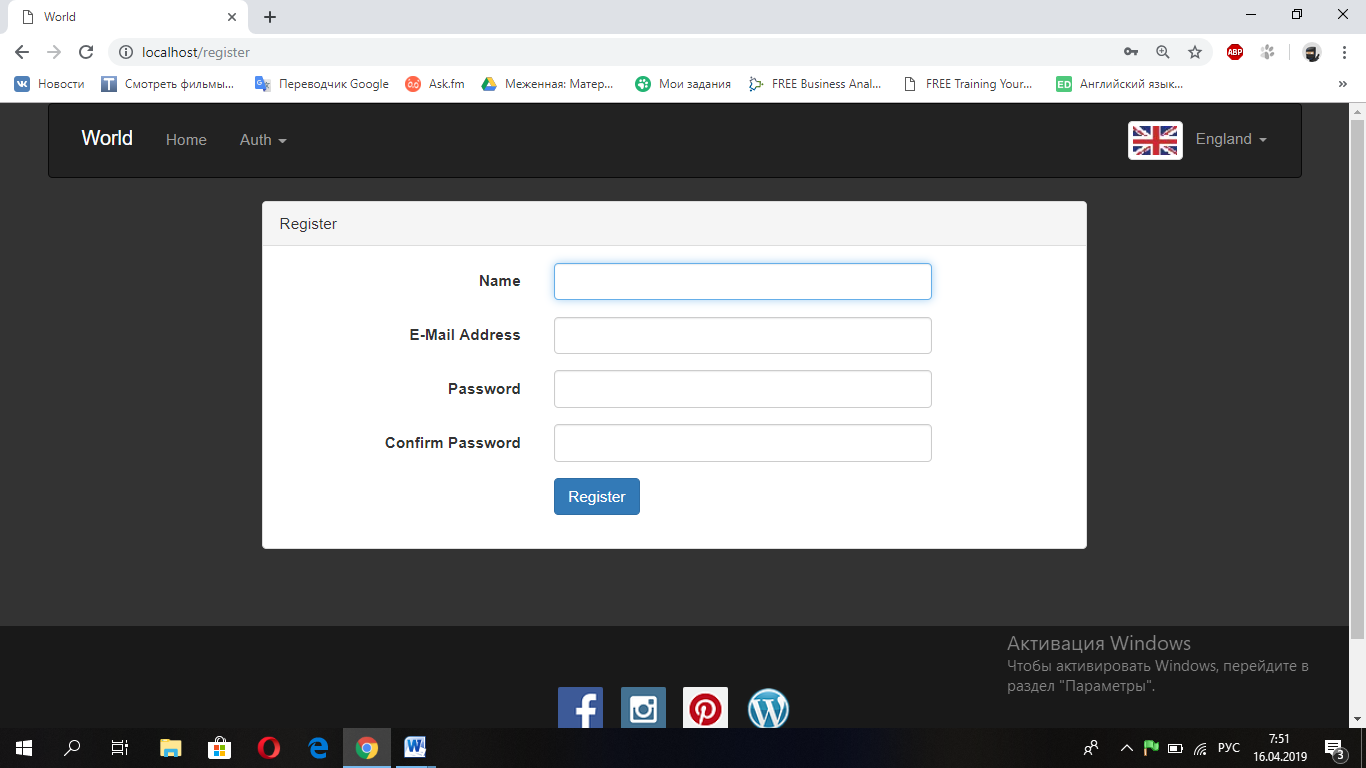


Рисунок 14 – Окно регистрации

После заполнения полей для логина и пароля в окне «Логин через  
e-mail» пользователь попадает в личный кабинет.

Навигация по сайту довольно проста и интуитивно понятна пользователю.

**ЗАКЛЮЧЕНИЕ**

В современном мире создание географических веб-приложений играет огромную роль на IT-пространстве. Эта необходимость связана с развитием информационных технологий, которые обусловлены ростом объема информации, а также значением Интернета в жизни человека. В настоящее время, использование различных географических сайтов, приложений, порталов распространено повсеместно и плотно вошло в повседневную жизнь пользователей.

Геопортал – это ресурс, в котором реализован симбиоз географических показателей, видеоконтента, важной информации и последних новостей. Помимо этого, на сайте можно авторизироваться и сохранять раннее открытые новости.

Геопортал позволяет получить данные о границах административно-территориальных единиц, а также предоставляет доступ к актуальным пространственным данным всем заинтересованным лицам.

Задача была реализована при использовании PHP-фреймворка, шаблона MVC, СУБД MySQL, языка разметки html, языка описания внешнего вида CSS и языка сценариев JavaScript. Также проект был занесён на онлайн-репозиторий GitHub.

Пользователи лишаются не нужных расходов на покупку различных бумажных носителей, имеют постоянный доступ к геопространственным данным и последней информации. На сайте интегрирована интерактивная карта, которая позволяет найти нужный регион или город. Регулярное обновление геопространственных данных и информации позволяет быть в курсе последних событий, а использование любого интернет-браузера делает геопортал доступным для пользователей с различными гаджетами.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] HTMLи CSS. Разработка и создание веб-сайтов/ Д. Дакетт

[2] JAVASCRIPT и JQuery.Интерактивная веб-разработка/ Д. Дакет

[3] Статьи о различных технологиях программирования [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/

[4] Основы Git [Электронный ресурс]. – Режим доступа: https://git-scm.com/

[5] Введение в MVC и HMVC [Электронный ресурс]. – Режим доступа: https://ruseller.com/

[6] Основы баз данных [Электронный ресурс]. – Режим доступа: https://www.site-do.ru/db/db1.php

[7] Создаём динамические веб-сайты с помощью PHP, MySQL, JavaScript, HTML5 и CSS / Р. Никсон - Издательство «Питер», 2016, - 768 с.

# ПРИЛОЖЕНИЕ А

Листинг кода

<?php

namespace App\Http\Controllers;

use App\Parser\GoogleNews;

use App\Parser\WikipediaInfo;

use App\Parser\Youtube;

use App\City;

use App\Video;

use App\News;

use App\CountryEn;

use App\CityGeo;

class AjaxController extends Controller

{

public function getNews(){

$country=$\_GET["country"];

$day = date('Y-m-d');

$ob = News::where('country\_name', $country)->where('putday', $day)->first();

if(!$ob){

$obj=new GoogleNews;

$pars=$obj->getParse($country);

$obn = new News;

$obn->country\_name = $country;

$obn->body = $pars;

$obn->lang = 'en';

$obn->putday = $day;

$obn->url = $\_SERVER['SERVER\_NAME'].$\_SERVER['REQUEST\_URI'];

$obn->save();

$parse = $pars;

}else{

$parse = $ob->body;

}

return view('ajax.iframe',compact("country", "parse"));

}

public function getInfo(){

$country=$\_GET["country"];

$obj=new WikipediaInfo;

$pars=$obj->getParse($country);

return view('ajax.info',compact("country"));

}

public function getCities(){

$country=$\_GET["country"];

$obj=CountryEn::where("country\_name",$country)->first();

$cities=CityGeo::where("country\_id",$obj->id)->get();

return view('ajax.city',compact("country","cities"));

}

public function getCabinet(){

$id=(int)$\_POST['id'];

$cities=CityGeo::where('country\_id',$id)->get();

return view('ajax.cabinet', compact("cities"));

}

public function getVideo(){

$country=$\_GET["country"];

$obj=new Youtube;

$pars=$obj->getParse($country);

echo $pars;

// return view('ajax.info',compact("country"));

}

}

@extends('layouts.app')

@section('content')

<div id="container"></div>

<div class="container fly">

<div class="col-md-10 col-md-offset-2">

<div class="panel panel-default">

<div class="panel-heading">

<a href="{{asset($obj->country->country\_iso\_code.'/'.$obj->country->country\_name)}}">

{{$obj->country->country\_name}}

</a>

{{$obj->name}}

</div>

<div class="panel-body">

<iframe width="100%" height="500" frameborder="0" scrolling="no" marginheight="0" marginwidth="0" src="https://www.openstreetmap.org/export/embed.html?bbox={{$obj->country->box}}&amp;layer=mapnik&amp;marker={{$obj->latitude}}%2C{{$obj->longitude}}" style="border: 1px solid black"></iframe><br/><small><a href="https://www.openstreetmap.org/?mlat={{$obj->latitude}}&amp;mlon={{$obj->longitude}}#map=8/53.907/27.564">Посмотреть более крупную карту\\//</a></small>

</div>

</div>

</div>

</div>

@endsection

<script src="{{ asset('js/app.js') }}"></script>

</head>

<body>

<div class="container">

@include('templates.nav')

<div class="col-md-12" id="logofon">

<div class="cont">

<div id="container" style="width: 100%"></div>

<br style="clear:both"/>

</div>

</div>

</div>

@yield('content')

<div class="footer">

<div class="social">

<a href="#"> <img src="{{asset('img/pic\_facebook.png')}}"/></a>

<a href="#"> <img src="{{asset('img/pic\_instogramm.png')}}"/></a>

<a href="#"> <img src="{{asset('img/pic\_pinterest.png')}}"/></a>

<a href="#"> <img src="{{asset('img/pic\_worldpress.png')}}"/></a>

</div>

&copy; <a href="http://mikhalkevich.colony.by" target="\_blank">mikhalkevich</a>

</div>

<!-- Scripts -->

<script src="{{ asset('js/lang.js') }}"></script>

@section('scripts')

@show

</body>

</html>

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\News;

class NewsController extends Controller

{

public function getAll(){

$news = News::orderBy('id','DESC')->paginate(50);

return view('news', compact('news'));

}

public function getOne($id = null){

$one = News::find($id);

return view('new\_one', compact('one'));

}

}

$(function () {

var country = $("#info").attr("data-country");

$.ajax({

url: '/ajax/info',

data: 'find=info&country=' + country,

success: function (data) {

console.log(data);

$("#empty\_info").html(data);

},

error: function (data) {

console.log(data);

}

});

$('#link\_news').click(function () {

var country = $(this).attr("data-country");

$.ajax({

url: '/ajax',

data: 'find=news&country=' + country,

success: function (data) {

$('#news').html(data);

},

error: function (data) {

$('#news').html(data);

}

});

});

$('#link\_cities').click(function () {

var country = $(this).attr("data-country");

$.ajax({

url: '/ajax/cities',

data: 'find=cities&country=' + country,

success: function (data) {

$('#cities').html(data);

},

error: function (data) {

$('#cities').html(data);

}

});

});

$('#video').click(function () {

var country = $(this).attr("data-country");

$.ajax({

url: '/ajax/video',

data: 'find=video&country=' + country,

success: function (data) {

console.log('ok');

$('#videos').html(data);

},

error: function (data) {

$('#videos').html(data);

}

});

});

});

Route::get('/', 'BaseController@getIndex')->middleware('cookie');

Route::get('news/','NewsController@getAll');

Route::get('news/{id}','NewsController@getOne');

Auth::routes();

Route::get('ajax', 'AjaxController@getNews');

Route::get('ajax/info', 'AjaxController@getInfo');

Route::post('ajax/cabinet', 'AjaxController@getCabinet');

Route::get('ajax/cities', 'AjaxController@getCities');

Route::get('ajax/video', 'AjaxController@getVideo');

Route::post('cabinet','CabinetController@postIndex');

Route::get('home', 'HomeController@index')->name('home');

Route::get('city/{id}','CityController@getCity');

Route::get('/cabinet', 'CabinetController@getIndex');

Route::get('{url}', 'CountryController@getIndex')->where('url', '[A-Za-z]{2}');

Route::get('{url}/{name}', 'CountryController@getName')->where('url', '[A-Za-z]{2,3}')->middleware('cookie');

<script>

$(function(){

$("#siteSub").hide();

$("#contentSub").hide();

$(".dablink").hide();

$(".mw-jump-link").hide();

$(".mbox-text-div").hide();

$(".toc").hide();

$(".toctitle").hide();

$(".mw-editsection").hide();

$(".noprint").hide();

</script>

<html lang="{{ app()->getLocale() }}">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta http-equiv="Cache-Control" content="no-cache">

<!-- CSRF Token -->

<meta name="csrf-token" content="{{ csrf\_token() }}">

<title>{{ config('app.name', 'Map of Worlds') }}</title>

<!-- Styles -->

<link href="{{ asset('bootstrap/css/bootstrap.min.css') }}" rel="stylesheet">

<link href="{{ asset('css/main.css') }}" rel="stylesheet">

@section('styles')

@show

<script src="{{ asset('js/app.js') }}"></script>

</head>

<body>

<div class="container">

@include('templates.nav')

<div class="col-md-12" id="logofon">

<div class="cont">

<div id="container" style="width: 100%"></div>

<br style="clear:both"/>

</div>

</div>

</div>

@yield('content')

<div class="footer">

<div class="social">

<a href="#"> <img src="{{asset('img/pic\_facebook.png')}}"/></a>

<a href="#"> <img src="{{asset('img/pic\_instogramm.png')}}"/></a>

<a href="#"> <img src="{{asset('img/pic\_pinterest.png')}}"/></a>

<a href="#"> <img src="{{asset('img/pic\_worldpress.png')}}"/></a>

</div>

&copy; <a href="http://mikhalkevich.colony.by" target="\_blank">mikhalkevich</a>

</div>

<!-- Scripts -->

<script src="{{ asset('js/lang.js') }}"></script>

@section('scripts')

@show

</body>

</html>

@extends('layouts.app')

@section('styles')

@parent

@endsection

@section('scripts')

@parent

<script src="{{asset('public/plugins/datamaps/d3.min.js')}}"></script>

<script src="{{asset('public/plugins/datamaps/topojson.min.js')}}"></script>

<script src="{{asset('public/plugins/datamaps/datamaps.world.min.js')}}"></script>

<script src="{{asset('public/js/my.js')}}"></script>

<script>

var map = new Datamap({

element: document.getElementById('container'),

geographyConfig: {

highlightBorderColor: '#bada55',

popupTemplate: function(geography, data) {

$('path').click(function () {

url = $(this).attr('class');

var res = url.split(" ");

document.location.href = '/' + res[1] + '/' + geography.properties.name;

});

return '<div class="hoverinfo">' + geography.properties.name + '</div>'

},

highlightBorderWidth: 3

}

});

</script>

@endsection

@section('content')

<div id="container"></div>

<div class="container">

<div class="row">

<div class="col-md-12 bodycon">

<div class="panel-body main\_body">

<div class="main\_page">

<!-- Nav tabs -->

<ul class="nav nav-tabs nav-mama" role="tablist">

<li role="presentation" class="active"><a href="#home" aria-controls="home" role="tab"

data-toggle="tab">News</a></li>

<li role="presentation"><a href="#services" aria-controls="calculate" role="tab"

data-toggle="tab">Photos</a></li>

<li role="presentation"><a href="#profile" aria-controls="profile" role="tab"

data-toggle="tab">Videos</a></li>

<li role="presentation"><a href="#messages" aria-controls="messages" role="tab"

data-toggle="tab">Travels</a></li>

</ul>

<!-- Tab panes -->

<div class="tab-content">

<div role="tabpanel" class="tab-pane active" id="home">

@foreach($news as $one)

<div>

<a href="{{asset('news/'.$one->id)}}">{{$one->putday}} - {{$one->country\_name}}</a>

</div>

@endforeach

</div>

<div role="tabpanel" class="tab-pane" id="services">

Countries

<br style="clear: both;">

<div class="text">

</div>

</div>

<div role="tabpanel" class="tab-pane" id="profile">

Cities

</div>

<div role="tabpanel" class="tab-pane" id="messages">

Travels

</div>

</div>

</div>

</div>

<h2 class="hdtext">Hottels near from here</h2>

<div class="picblock">

Hottels

</div>

</div>

</div>

</div>

@endsection