Введение (задачи, актуальность, существующие решения)

Глава 1. Геоинформационные системы и веб-технологии

1.1. Геоинформационные системы

1.2. Веб-технологии в ГИС

1.3. Технологии баз данных в ГИС (описание структур данных и

геометрических типов данный в современных СУБД)

Глава 2. Проектирование платформы для создания геоинформационных

систем

2.1. Основные функции платформы ГИС (что мы планируем сделать)

2.2. Проектирование БД

2.3. Проектирование интерфейса системы

Глава 3. Реализация геоинформационной системы

3.1. Описание программных модулей

Заключение (результаты и их отличие от предшествующих решений)

Литература

**ВВЕДЕНИЕ**

Современные геоинформационные системы предоставляют достаточно широкие возможности по работе с геопространственными данными. Многие существующие платформы ГИС [1] обеспечивают работу в различных системах координат, в различных форматах геоданных. При этом поддерживаются различные картографические основы, в том числе и спутниковые снимки поверхности замели. Однако большинство современных геоинформационных систем достаточно сложны в использовании обычными пользователями и требуют определённых навыков работы.

Задачей исследования является изучение существующих платформ для разработки геоинформационных систем и построение собственной платформы, позволяющей создавать ГИС различного назначения с использованием веб технологий.

Разрабатываемая платформа для создания геоинформационных систем должна обеспечивать работу с данными в различных форматах, поддерживать различные картографические основы и поддерживать наиболее популярные системы управления базами данных для хранения геоданных, а также иметь простой и интуитивно понятный интерфейс.

В настоящее время российские и зарубежные производители предлагают широкий спектр геоинформационных систем. Можно привести наиболее популярные из них.

* GIS MapInfo – содержит всю информацию (графическую, текстовую и др.) в таблицах. Пользователь управляет процессами создания картографической информации и тематических слоев через графический интерфейс, который выражается в виде меню и системы контекстных подсказок. MapInfo имеет встроенную реляционную СУБД, работающую по SQL-запросам с файлами в различных форматах.
* BusinessMAP – пакет начального уровня для решения бизнес-задач, таких как поддержка продаж и маркетинг, на основе включенных в него (в основном по США) или уже имеющихся в готовом виде данных.
* Atlas GIS – удобный пакет для конечного пользователя со всеми стандартными функциями, имеющимися в простых настольных ГИС системах. Включает разнообразные готовые данные, в том числе по России. Основное назначение – бизнес-приложения.
* ArcView GIS – настольная ГИС с развитыми функциями пространственного анализа, многие из которых добавляются к базовому набору в виде дополнительных модулей. Работает на разных платформах.
* ARC/INFO – профессиональная ГИС для решения задач структурирования и управления данными, ведения базы данных. Работает на разных платформах. ГИС для рабочей группы (уровня отдела или небольшой фирмы, в которых постоянно ведется разработка ГИС проектов). По мере расширения сферы деятельности и усложнения выполняемых задач целесообразно распределить их решение между несколькими продуктами. Такая ГИС обычно базируется на разделяемой базе пространственных данных, поддерживаемой и обновляемой с помощью разных продуктов и обеспечивающей работу «конечных» пользователей на персональных компьютерах в пределах данной рабочей группы или отдела. Конкретное предлагаемое решение зависит от типов используемых данных, выбранной программной среды, аппаратного обеспечения и ряда других факторов. ГИС уровня предприятия (корпоративная ГИС). Обеспечивает всю организацию пространственной информационной поддержкой. В такую ГИС можно встроить сотни специализированных приложений с дополнительными средствами пространственных запросов, расширенного анализа, отображения и представления данных. В качестве компонентов системы такого уровня могут использоваться серверы пространственных данных:

**Глава 1. Геоинформационные системы и веб-технологии**

**1.1. Геоинформационные системы**

1.1.1 Описание геоинформационных систем

Геоинформационные системы (ГИС) – это высоко интегрированные системы, обеспечивающие доступ к анализу географической и геодезической информации. Геоинформационные системы могут использоваться для предоставления детальной картографии, сбора информации о ландшафте и окружающей среде, мониторинга и управления сценариями обслуживания клиентов, проработки систем автоматического присвоения адресов и др.

ГИС может представлять собой набор функционально связанных программ, простое веб-приложение или форматированную карту, содержащую географическую информацию, такую как растительность, географические границы, популяции и другие данные. Системы могут быть настроены таким образом, чтобы показывать аналитику, например популяционные плотности или мощность населения. Также могут быть добавлены и другие объекты, такие как наличие медицинского оборудования, дорог и даже архитектуры. ГИС позволяет пользователям исследовать, анализировать и делать заключения о географических данных и отображаемых изображениях в различных форматах.

Геоинформационные системы обычно классифицируются в зависимости от используемой технологии и их способа подключения к другим аппаратным компонентам. Обычно они группируются по типу данных, на основе которых они работают, – растровые и векторные данные.

Растровые данные представляют собой классическую дискретную графику, и они могут быть использованы для создания подробных карт территории.

Векторные системы характеризуются одномерными и двумерными данными, более общей внешней геометрией, предназначенными для получения интерактивных и детализированных карт и изображений. Кроме того, существует аналитический класс ГИС, который обеспечивает лучшую аналитику для географических данных. Аналитические ГИС обычно включают технологии текстового поиска, кластеризации, статистической мощности и маршрутных алгоритмов.

В ГИС данные представляются в виде трех начальных структур данных:

* Классы пространственных объектов;
* Атрибутивные таблицы;
* Растровые наборы данных.

*Классы пространственных объектов* – это наборы географических объектов с общим типом геометрии (точки, линии или полигоны) и общим набором атрибутов, которые можно отобразить на карте или использовать для анализа. Например, вы можете создать класс пространственных объектов для дорог, земельных участков или канализационных люков.

Для создания класса пространственных объектов в ГИС вам нужно задать несколько свойств, таких как имя, тип геометрии, система координат, поля и индексы. Классы пространственных объектов в ГИС могут быть связаны с другими таблицами или слоями по общему атрибуту. Классы пространственных объектов в ГИС также могут быть конвертированы из других источников данных, таких как шейп-файлы или данные САПР.

*Атрибутивные таблицы* – это таблицы, которые содержат информацию об атрибутах пространственных объектов, таких как точки, линии или полигоны. Атрибуты – это характеристики объектов, например, их название, тип, цвет или размер. Атрибутивные таблицы позволяют визуализировать, анализировать и редактировать данные о пространственных объектах.

Атрибутивные таблицы состоят из строк и столбцов. Каждая строка соответствует одному пространственному объекту, а каждый столбец – одному атрибуту. Атрибут в таблице также называется полем или колонкой. Каждое поле может иметь определенный тип данных, такой как число, текст или дата. Некоторые поля, такие как геометрия или уникальный идентификатор объекта, добавляются и поддерживаются автоматически.

Атрибутивные таблицы могут храниться в разных форматах данных, таких как файлы dBASE, текстовые файлы, базы данных или базы геоданных. Базы данных и базы геоданных могут поддерживать дополнительные типы данных и функции для таблиц, такие как BLOB [], растры, целостность данных и управление версиями.

Атрибутивные таблицы могут быть связаны с другими таблицами по общему полю, называемому ключевым. Это позволяет ассоциировать записи одной таблицы с записями в другой и получать доступ к связанной информации. Связи между таблицами могут быть временными или постоянными в зависимости от способа создания.

*Растровые наборы данных* – это наборы изображений или растровых данных, которые представляют пространство как массив равновеликих пикселов, каждый из которых имеет значение и координаты. Растровые наборы данных могут быть непрерывными или тематическими. Например, вы можете использовать растровые наборы данных для представления высоты, температуры, землепользования или отсканированных карт. Для хранения и управления растровыми наборами данных в ГИС вы можете использовать файловую систему, базу геоданных или набор данных мозаики. Набор данных мозаики – это совокупность растровых наборов данных, хранящаяся как каталог и отображаемая как единое мозаичное изображение или отдельные изображения. Растровые наборы данных в ГИС полезны для разных целей, таких как:

* Изображения как базовые карты. Растровые наборы данных могут использоваться в качестве фоновых изображений для других векторных слоев на карте, чтобы обеспечить пространственное выравнивание, контекстную информацию и визуальный интерес.
* Растры как карты поверхности. Растровые наборы данных могут использоваться для представления данных, которые непрерывно изменяются в ландшафте, таких как высота, осадки или температура. Растры позволяют хранить и анализировать непрерывные данные в виде поверхности.
* Растры как источники анализа. Растровые наборы данных могут использоваться для выполнения различных видов анализа, таких как классификация, интерполяция, фильтрация или алгебра растров. Результаты анализа могут быть сохранены как новые растровые наборы данных или отображены на карте.

ГИС используются для хранения, обработки и анализа географической информации. Они позволяют создавать карты и планы, анализировать пространственные данные, моделировать процессы и явления в пространстве, определять оптимальный маршрут и многое другое.

1.1.2. Математические аспекты картографии

**Картографическая проекция** – математически определённый способ отображения поверхности Земли (в общем смысле, любой искривлённой поверхности) на плоскость.

Суть проекций связана с тем, что фигуру небесного тела (для Земли – геоид, для простоты обычно считаемый эллипсоидом вращения), не развёртываемую в плоскость, заменяют на другую фигуру, развёртываемую на плоскость. При этом с эллипсоида на другую фигуру переносят сетку параллелей и меридианов. Вид этой сетки бывает разный в зависимости от того, какой фигурой заменяется эллипсоид.

В любой проекции существуют искажения, они бывают четырёх видов: *искажения длин; искажения углов; искажения площадей; искажения форм;*

**Искажение длин** – базовое искажение, из которого логически вытекают остальные искажения. Причиной тому является невозможность развернуть поверхность эллипсоида (или шара) на плоскости без складок или разрывов. Искажение длин означает непостоянство масштаба плоского изображения, что проявляется в изменении масштаба от точки к точке, и даже в одной и той же точке в зависимости от направления.

**Искажения площадей** логически вытекают из искажения длин. За характеристику искажения площадей принимают отклонение площади эллипса искажений от исходной площади на эллипсоиде.

**Искажения углов** логически вытекают из искажения длин. За характеристику искажений углов на карте принимают разность углов между направлениями на карте и соответствующими направлениями на поверхности эллипсоида.

**Искажения формы** – графическое изображение вытянутости эллипсоида.

Существует три семейства картографических проекций: конических, цилиндрических и плоскостных проекций.

***Коническая проекция –*** самая простая проекция. Проходит по касательной к глобусу вдоль линии широты. Эта линия называется стандартной параллелью. Меридианы проецируются на коническую поверхность, сходясь на вершине или в точке конуса. Параллели проецируются на коническую поверхность как кольца. Конус затем «рассекается» вдоль любого меридиана для создания конечной конической проекции, в которой имеются прямые сходящиеся меридианы и параллели, представленные концентрическими окружностями. Меридиан, противолежащий линии сечения, становится центральным меридианом.

Более сложные конические проекции соприкасаются с поверхностью глобуса в двух местах. Эти проекции называются секущими коническими проекциями и определяются двумя стандартными параллелями. Характер искажений при секущих проекциях различается для районов, расположенных между стандартными параллелями, и для районов, расположенных за их пределами. Как правило, секущая проекция дает меньшее суммарное искажение, чем касательная проекция. В еще более сложных конических проекциях ось конуса не совпадает с полярной осью глобуса. Такие проекции называются косыми.

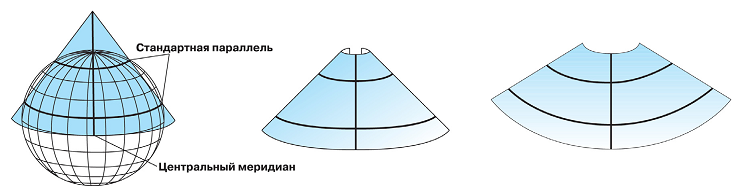


Рис. 1 – Конические проекции

В целом, чем дальше от стандартной параллели, тем больше искажение. Соответственно, отсечение верхушки конуса создает более точную проекцию. Этого можно достичь, если не использовать полярную область при проецировании объектов. Конические проекции используются для среднеширотных зон, имеющих ориентацию с востока на запад.

***Цилиндрические проекции*** могут также быть касательными или секущими. Проекция Меркатора [1] является одной из наиболее простых цилиндрических проекций, и экватор обычно является ее линией касания. Меридианы проецируются геометрически на цилиндрическую поверхность, а параллели проецируются математически.

При этом создается координатная сетка с углами 90°. Цилиндр «рассекается» вдоль любого меридиана для получения конечной цилиндрической проекции. Меридианы расположены через равные интервалы, в то время как интервал между параллельными линиями широты возрастает по направлению к полюсам. Эта проекция является равноугольной и показывает истинное направление вдоль прямых линий. В проекции Меркатора прямыми линиями являются линии румбов – линии постоянного азимута, а не большинство больших окружностей.

Во всех цилиндрических проекциях линия касания или линии сечения не имеют искажений, и, таким образом, являются линиями равных расстояний. Другие географические свойства варьируют в зависимости от конкретной проекции.

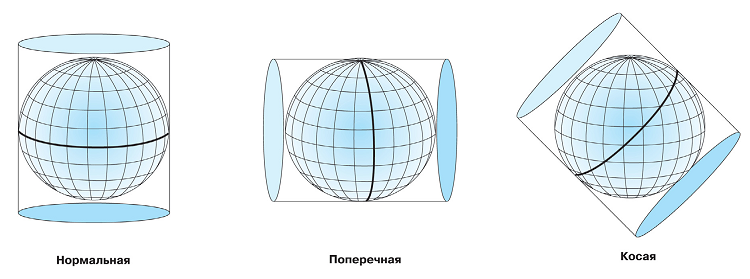


Рис. 2 – Цилиндрические проекции

***Проекции на плоскость –*** проецируют картографические данные на плоскую поверхность, касающуюся глобуса. Проекция на плоскость также известна также как азимутальная или зенитная проекция. Этот вид проекции обычно идет по касательной к глобусу в одной точке, но может быть и секущим. Точкой контакта может быть Северный полюс, Южный полюс, точка на экваторе или любая точка между ними. Эта точка определяет используемую ориентировку и является фокусом проекции. Фокус определяется центральной долготой и центральной широтой. Ориентировка проекций может быть полярной (нормальной), экваториальной (поперечной) и косой.



Рис. 3 – Проекции на плоскость

Полярные проекции представляют собой простейшую форму этого вида проекций. Параллели широты отходят от полюса как концентрические окружности, а меридианы представлены прямыми линиями, которые пересекаются на полюсе под своими истинными углами. При всех остальных ориентировках проекции на плоскость будут иметь углы координатной сетки 90° в своем центральном фокусе. Направления из фокуса являются точными.

***Точность картографических проекций*.**

Картографические проекции по определению не могут передать сферическую поверхность со 100% точностью. В ходе проецирования любая карта будет иметь искажения углов, расстояний или площадей. Картографическая проекция может быть компромиссной, т.е. искажать все три свойства в некоторых допустимых пределах.

По характеру искажений, возникающих при переходе от сферической поверхности к плоскости, картографические проекции подразделяют на *равновеликие* (сохраняют размеры площадей), *равноугольные* (оставляют углы и формы контуров без искажений) и *равнопромежуточные* проекции (про­ек­ции, в ко­то­рых мас­штаб по­стоя­нен по од­но­му из главных на­прав­ле­ний: по ме­ри­диа­ну или па­рал­ле­ли).

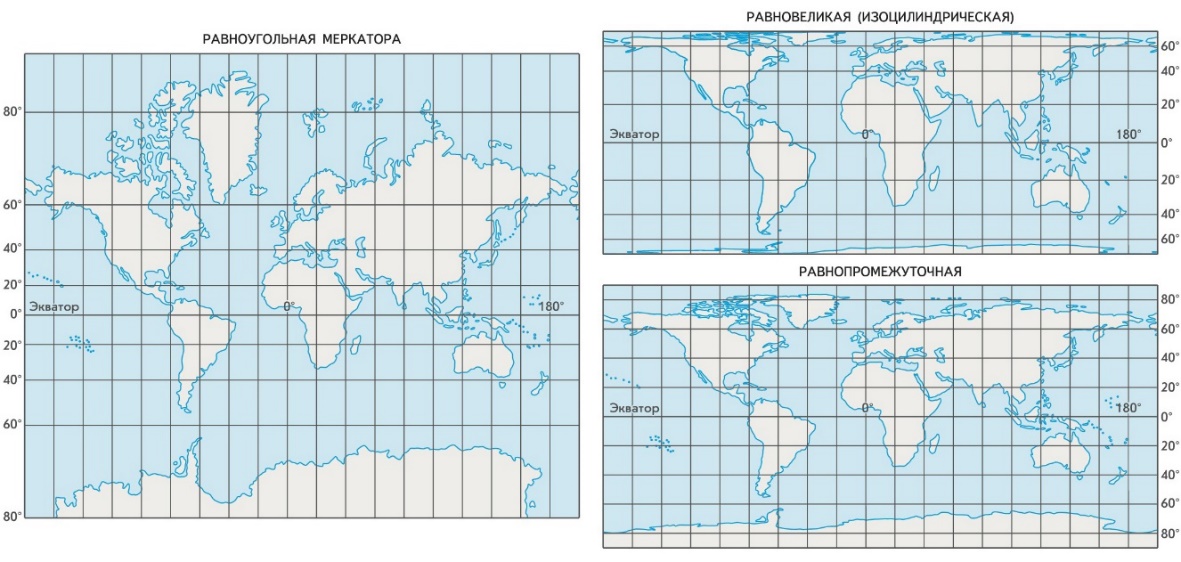


Рис. 4 – Типы искажений

**Системы координат, используемые в ГИС.**

Системы координат позволяют использовать географические данные общей локализации для их интеграции. Системы координат используются для локализации географических объектов, изображений и точек наблюдений, таких, как GPS-позиции на общегеографической сетке.

В геодезических приложениях применяются общеземные (глобальные) и референцные (региональные) системы координат, а также небесная система координат. К референцным системам координат относятся национальные геодезические системы координат.

**Общеземные системы координат.**

По определению общеземная система координат является геоцентрической пространственной системой координат с началом в центре масс Земли (Рис. 5). Ось Z направлена к условному земному полюсу (международному условному началу), как определено рекомендациями Международной службы вращения Земли (IERS), ось Х – в точку пересечения плоскости экватора и начального меридиана, установленного IERS и Международным бюро времени (BIH), ось Y дополняет систему до правой. Общеземная система координат вращается вместе с Землей.

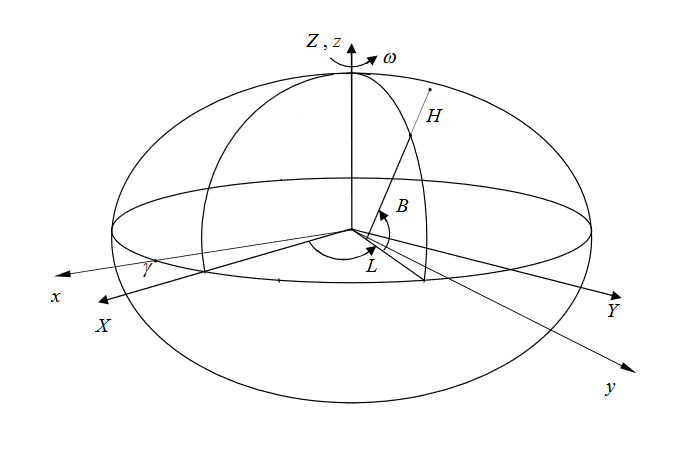


Рис. 5 – Система координат

В геоцентрической системе координат положение точки в пространстве определяется значениями координат X, Y, Z. В геодезических приложениях для этой же цели используются геодезические координаты B, L, H, относящиеся к общеземному эллипсоиду – эллипсоиду вращения, геометрический центр которого совпадает с центром общеземной системы координат (Рис. 5). Ось Z является осью вращения эллипсоида.

Геодезическая широта B определяется как угол между нормалью к эллипсоиду, проходящей через заданную точку, и плоскостью экватора; геодезическая долгота L – двугранный угол между плоскостью начального меридиана и плоскостью меридиана, проходящего через заданную точку (положительное направление счета долгот от начального меридиана к востоку, от 0° до 360°); геодезическая высота H – отрезок нормали к общеземному эллипсоиду от его поверхности до заданной точки.

***Референцные геодезические системы координат.***

Отсчетной поверхностью в геодезических референцных системах координат, как и в общеземной системе координат, является эллипсоид вращения. Отличие заключается в том, что положение начала счета координат в этих системах и ориентировка их осей могут не совпадать. При пересчете геодезических координат B, L, H в пространственные прямоугольные X, Y, Z в референцной системе используются те же соотношения, что и в общеземной геодезической системе координат.

***Системы координат, используемые на практике.***

Практическими реализациями пространственной геоцентрической земной системы координат являются системы координат WGS-84, ПЗ-90.11 и ГСК-2011.

***Система координат WGS-84*** (World Geodetic System – Всемирная геодезическая система координат) – это система геодезических параметров Земли 1984 года, используемая в GPS, в число которых входит система геоцентрических координат). WGS-84 определяет координаты относительно центра масс Земли, погрешность составляет менее 2 см. В WGS 84 нулевым меридианом считается Опорный меридиан, проходящий в 5,31″ (~100 м) к востоку от Гринвичского меридиана [2]. За основу взят эллипсоид с бóльшим радиусом — 6 378 137 м (экваториальный) и меньшим – 6 356 752, 3142 м (полярный). Практическая реализация идентична отсчётной основе ITRF. С 1997 года WGS84 поддерживается в пределах 10 см в эпохе до 2005 года, в то время как в ITRF действует текущая эпоха ITRF2014.

***Система координат ПЗ-90.11*** [3] (общеземная геоцентрическая система координат «Параметры Земли 1990 года») — это государственная система координат, используемая в ГЛОНАСС. Начало системы расположено в центре масс Земли и характеризуется средними квадратическими погрешностями 1-2 м.

ПЗ-90.11 была установлена постановлением Правительства РФ от 24 ноября 2016 года №1240 для использования в целях геодезического обеспечения орбитальных полетов, решения навигационных задач и выполнения геодезических и картографических работ в интересах обороны Российской Федерации.

***Система координат ГСК-2011*** (геодезическая система координат 2011 года) – это государственная система координат, установленная постановлением Правительства РФ от 24 ноября 2016 года №1240 для использования при осуществлении геодезических и картографических работ на территории Российской Федерации.

Оптимальной системой координат при разработке веб платформы для создания геоинформационных систем является WGS-84, поскольку является мировым стандартом и обеспечивает необходимую точность и удобство использования при взаимодействии с ГИС.

**1.2. Веб-технологии в ГИС**

Веб-технологии позволяют быстро и просто передавать карты и данные из одного формата в другой. Такие приложения позволяют создавать слои карт и позволяют людям работать с данными с любого места на Земле. Кроме того, люди используют веб-технологии для создания виртуальных карт и просмотра самых новых данных. Они имеют мощный инструментарий, позволяющий создавать легко доступные онлайн-приложения для анализа данных и просмотра карт. С помощью веб-технологий можно создавать надежные и безопасные веб-карты и интерактивные интерфейсы.

В качестве одного из языков программирования был выбран язык PHP [2], скриптовый язык программирования с открытым исходным кодом. Этот язык программирования позволяет использовать современные концепции, технологии и различные библиотеки, упрощающие разработку приложений. Язык PHP отлично подходит для разработки клиент-серверных приложений, поскольку обладает рядом неоспоримых преимуществ, среди которых:

* Высокая скорость работы, что обеспечивает общую высокую производительность;
* Открытый исходный код, что обеспечивает обширную поддержку от сообщества, которое улучшает и дополняет функционал языка;
* Простой в освоении синтаксис, низкий порог вхождения;
* Отличная совместимость и переносимость – приложения работают хорошо на разных платформах, что является ключевым аспектом при создании кроссплатформенных приложений. Поддержка большинства серверов (например, Apache, Netscape и т.д.) и более 20 баз данных (включая MySQL, PostgreSQL и другие);
* Большая пластичность, емкость и работоспособность. PHP-приложения можно проектировать отдельно без привязки к разработке веб-страницы;
* Хорошая масштабируемость проектов, созданных при помощи языка PHP;

Ещё одним языком программирования был выбран JavaScript [3] – язык программирования, поддерживающий несколько парадигм программирования: объектно-ориентированный, императивный и функциональный стили. В первую очередь применяется для добавления интерактивности в веб-страницах. JavaScript обладает большим количеством фреймворков (Angular, React, Node.js и т.д.), которые упрощают разработку приложений. К основным достоинствам можно отнести:

* Высокая скорость и производительность. JavaScript позволяет обрабатывать код на устройствах пользователей, что значительно уменьшает нагрузку на сервер и экономит Интернет-трафик;
* Отличная интеграция с веб-технологиями. Все популярные браузеры поддерживают скрипты, написанные на JavaScript. Интеграция с серверными частями приложений (backend);
* Простой в освоении синтаксис, низкий порог вхождения;
* Развитая инфраструктура. Для JS существует множество различных библиотек и фреймворков, например Angular, React, Node.js, Underscore и т.д.

**OpenLayers** – позволяет очень быстро и легко создать web-интерфейс для отображения картографических материалов, представленных в различных форматах и расположенных на различных серверах. Благодаря OpenLayers разработчик имеет возможность создать, к примеру, собственную карту, включающую слои, предоставляемые WMS (и WFS) серверами, такими как Mapserver, ArcIMS или Geoserver, и данными картографических сервисов Google. Библиотека является разработкой с открытым исходным кодом. OpenLayers обладает следующими возможностями:

* Добавления на карту панели навигации (включена по умолчанию). На панели находятся кнопки сдвига карты (север-юг, запад-восток), увеличения и уменьшения масштаба;
* Сдвига карты при помощи мыши;
* Изменения масштаба карты при прокрутке среднего колеса мыши;
* Получения координат точки, над которой находятся указатель мыши;
* Добавления панели управления видимостью/невидимостью слоев карты;
* Выбора произвольного объекта и получения атрибутивной информации о нем;
* Управления прозрачностью используемых слоев карты;
* Добавления к карте определяемых пользователем элементов (точек, линий, полигонов);

**Leaflet** – это библиотека с открытым исходным кодом на JavaScript, предназначенная для отображения карт на веб-сайтах. Она поддерживает большинство мобильных и стационарных платформ из числа современных браузеров. Библиотека Leaflet допускает подключение через CDN, устанавливать JavaScript-библиотеку не нужно. Библиотека поддерживает слои Web Map Service (WMS), GeoJSON, векторные и тайловые слои. Многие другие типы слоев поддерживаются дополнительными модулями. Leaflet обладает практически идентичными возможностями, что и OpenLayers, однако имеет не такую удобную документацию, как её аналог.

Для работы с картами была выбрана библиотека OpenLayers. Ключевым аспектом выбора являлся тот факт, что Leaflet разработан украинским программистом, поэтому не известно, насколько стабильно будет работать библиотека с учётом текущего положения в мире.

При разработке прототипа приложения были использованы веб-технологии JQuery и AJAX. Ниже каждая из технологий рассматривается подробнее.

**JQuery** – набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека JQuery помогает легко получать доступ к любому элементу DOM [2], обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека JQuery предоставляет удобный API для работы с AJAX. Также существует JQuery Mobile – сенсорно-ориентированный веб-фреймворк, направленный на кросс-браузерность с уклоном в сторону смартфонов и планшетов.

К основным возможностям JQuery можно отнести:

* Движок кросс-браузерных CSS-селекторов Sizzle, выделившийся в отдельный проект;
* Переход по дереву DOM, включая поддержку XPath как плагина;
* Обработка любых событий, возникающих на странице, такие как перемещение курсора мыши, нажати кнопок на клавиатуре или загрузка документа. Эти методы позволяют работать как со стандартными событиях Javascript, так и с событиями, предоставленными самой библиотекой JQuery.
* Визуальные эффекты. К базовым эффектам в JQuery относятся эффекты скрытия и отображения элементов, которые достигаются с помощью методов show(), hide() и toggle().Так, например, мы можем скрывать и отображать элементы по клику по кнопке ;
* AJAX-дополнения;
* JavaScript-плагины – модули, упрощающие разработку веб-приложений.

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчика события нажатия кнопки управление передаётся JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика. Библиотека содержит функциональность, полезную для максимально широкого круга задач.

**AJAX** (от англ. Asynchronous Javascript and XML – «асинхронный JavaScript и XML») – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

AJAX [3] – не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

* использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например с использованием XMLHttpRequest [3] (основной объект);
  + через динамическое создание дочерних фреймов;
  + через динамическое создание тега <script>;
  + через динамическое создание тега <img>.
* использование DHTML [4] для динамического изменения содержания страницы;

Действия с интерфейсом преобразуются в операции с элементами DOM (Document Object Model), с помощью которых обрабатываются данные, доступные пользователю, в результате чего представление их изменяется. Здесь же производится обработка перемещений и щелчков мышью, а также нажатий клавиш. Каскадные таблицы стилей, или CSS (Cascading Style Sheets), обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам. Объект XMLHttpRequest (или подобные механизмы) используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных.

Для лучшего понимания проведём сравнение стандартного подхода и подхода с использованием AJAX:

В классической модели веб-приложения:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. Браузер формирует и отправляет запрос серверу;
3. В ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д., после чего браузер полностью перезагружает всю страницу.

При использовании AJAX:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. Скрипт (на языке JavaScript) определяет, какая информация необходима для обновления страницы;
3. Браузер отправляет соответствующий запрос на сервер;
4. Сервер возвращает только ту часть документа, на которую пришёл запрос;
5. Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы).

Среди преимуществ подхода с использованием AJAX-технологии, можно выделить несколько основных:

1. **Экономия трафика, уменьшение нагрузки на сервер**. При правильной реализации AJAX позволяет снизить нагрузку на сервер в несколько раз. В частности, все страницы сайта чаще всего генерируются по одному шаблону, включая неизменные элементы («шапка», «навигационная панель», «подвал» и т. д.), для генерации которых требуются обращения к разным файлам, время на обработку скриптов (а иногда и запросы к БД) – всё это можно опустить, если заменить полную загрузку страницы генерацией и передачей лишь содержательной части;
2. **Ускорение реакции интерфейса**. Поскольку загрузка изменившейся части значительно быстрее, то пользователь видит результат своих действий быстрее и без мерцания страницы (возникающего при полной перезагрузке);
3. **Возможности для интерактивной обработки**. Например, при вводе поискового запроса в Google выводится подсказка с возможными вариантами запроса. На многих сайтах при регистрации пользователь вводит имя и сразу же видит, доступно это имя или нет. AJAX удобен для программирования чатов, административных панелей и других инструментов, которые выводят меняющиеся со временем данные;
4. **Мультимедиа не останавливается**. Поскольку страница не перезагружается, плеер продолжает работать. Это полезно для аудио- и видеохостингов.

**1.3. Технологии баз данных в ГИС**

Системы геоинформационных технологий (ГИС) используют базы данных для хранения и манипулирования данными о локациях, сущностях, дорогах, границах и других географических характеристиках. Базы данных используются для хранения многих географических объектов, таких как слои карт и наборы данных для анализа. Они также могут поддерживать слои данных для аналитики и действий в промежутках человеческих правок. Базы данных ГИС включают в себя различные программное обеспечение, такое как векторные приложения и базы данных изображений. Эти базы данных поддерживают функциональность, которая позволяет легко получать, манипулировать и хранить данные.

Одной из важнейших функций современных ГИС-приложений является возможность сохранения геометрических объектов, нанесённых на картографическую проекцию.

**Типы классов пространственных объектов.**

Векторные объекты (географические объекты с векторной геометрией) разносторонние и являются часто используемыми географическими типами данных, хорошо подходящими для представления объектов с дискретными границами, например улицы, административные границы и земельные участки. Пространственный объект – это объект, который хранит свое географическое представление, представленное обычно в виде точки, линии или полигона, в качестве одного из свойств (полей) в строке. Классы пространственных данных – однотипные совокупности объектов с общим пространственным представлением и набором атрибутов, хранящиеся в таблице базы данных, например, класс линейных объектов, представляющий осевые линии дорог.

Создавая класс пространственных объектов, необходимо задать тип объектов для определения типа класса пространственных объектов (точка, линия, полигон и т.д.).

Как правило, классы пространственных объектов являются тематическими наборами точек, линий или полигонов, но в действительности существует семь типов классов пространственных объектов. Первые три поддерживаются в базах данных и базах геоданных. Остальные четыре поддерживаются только в базах геоданных.

Каждый класс пространственных объектов – это собрание географических объектов с одинаковым типом геометрии (точка, линия или полигон), с однотипными атрибутами и одинаковой пространственной привязкой.

* **Точки**: пространственные объекты, которые слишком малы, чтобы обозначать их линиями или полигонами, а также точечные местоположения (точки GPS);
* **Линии**: отображают форму и местоположение географических объектов, слишком узких для отображения в виде полигонов (центральные линии улиц, ручьи). Линии также используются для представления объектов, имеющих длину и не имеющих площади, таких как изолинии и границы;
* **Полигоны**: набор многосторонних площадных объектов, представляющих форму и местоположение однородных типов пространственных объектов, таких как административные районы, округа, участки земли, типы почвы и зоны землепользования;
* **Аннотации**: подпись на карте, содержащая параметры отображения текста. Например, помимо текстовой строки каждой аннотации, там хранятся и другие свойства, такие как точки фигуры для размещения текста, его шрифт и размер в точках, а также другие свойства отображения. Аннотация может также быть связанной с надписываемыми объектами и может содержать подклассы;
* **Объекты-размеры**: специальный тип аннотации, показывающий специфические длины или расстояния, например, для указания длины стороны здания, участка земли или расстояния между двумя объектами. Размеры чаще всего используются для дизайнерских и инженерных задач в ГИС;

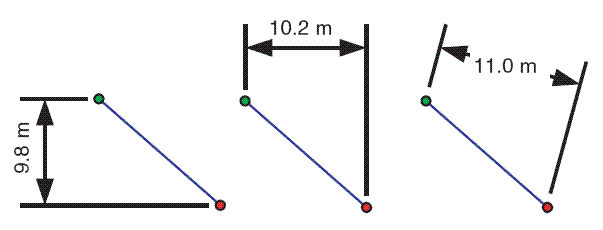


Рис. 1 – Пример объектов-размеров

* Мультиточки: пространственные объекты, состоящие из более чем одной точки. Мультиточки часто используются для управления массивами очень больших совокупностей точек, таких как, например, кластеры точек LiDAR, которые могут содержать миллиарды пунктов. Использование одной записи для такой точечной геометрии недопустимо. Объединение таких данных в группы записей объектов-мультиточек предоставляет возможность базе геоданных управлять массивными наборами точек.

Четыре основных типа классов пространственных данных – это точки, линии, полигоны и аннотации (названия для подписей на картах).

На рисунке ниже они представлены четырьмя наборами данных, относящимися к одной области: (1) – местоположения крышек люков в виде точек, (2) – линейные канализационные трубы, (3) – полигональные земельные участки и (4) – аннотации названий улиц.

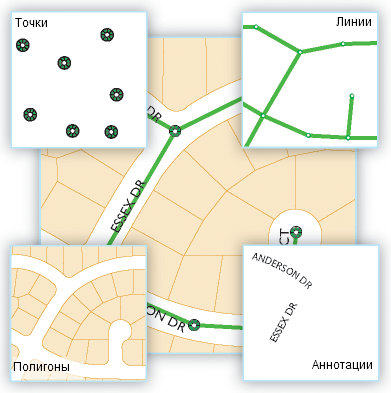


Рис. 2 – Пространственные данные

**Глава 2. Проектирование платформы для создания геоинформационных систем**

**2.1. Основные функции платформы ГИС**

**2.2. Проектирование базы данных**

Исследование географической информационной системы общего характера показало, что основными сущностями рассматриваемой предметной области являются: Пользователи системы, проекты, слои и объекты.

Рассмотрим свойства пользователей:

**Пользователи** системы имеют следующие свойства:

* Имя – имя пользователя (в некоторых случаях возможно: Фамилия, Имя, Отчество);
* Логин – идентифицирующее свойство пользователя;
* Пароль – аутентифицирующее свойство пользователя;
* Роль – свойство, определяющее его функциональные возможности в системе;
* Почта – свойство для контактов с пользователем по электронной почте.

**Проект** – это отдельная ГИС, реализованная на разрабатываемой платформе и определяемая некоторой областью применения.

Проект имеет следующие свойства:

* Наименование проекта;
* Автор проекта;
* Дата начала разработки;
* Дата последней модификации;
* Область применения – сфера деятельности, для автоматизации которой разрабатывается данный проект.

**Слой** – контейнер для размещения объектов

Слой обладает следующими свойствами:

* Наименование слоя.

**Объект** — объект учёта в ГИС, который размещен на слоях проектов

Свойства объекта:

* Наименование объекта;
* Тип объекта – необходим для поиска и фильтрации подобных объектов
* Координаты объекта – объект GeoJSON;
* Графический примитив – тип графического примитива, представляющий его на карте;
* Иконка – графическое обозначение объекта (например, ссылка на графический файл);
* Длина – максимальная длина объекта (если применимо);
* Ширина – максимальная ширина объекта (если применимо);
* Высота – максимальная высота объекта (если применимо);
* Площадь – площадь объекта (если применимо);
* Объём – объем объекта (если применимо)

Для сущности объект перечень его свойств не является окончательным, поскольку разрабатываемая платформа предполагает возможность для разработчиков ГИС создание собственных объектов, которые могут иметь дополнительные свойства (например объект школа может иметь множество свойств общеобразовательной школы: номер, тип школы, кол-во учеников и пр.).

Между рассмотренными сущностями имеются следующие отношения (связи):

* **Проект — Пользователь** – отношение обозначающее участие пользователя в проекте. Данное отношение является отношением многие ко многим, так как несколько пользователей могут участвовать в оном проекте, и один пользователь может участвовать в нескольких проектах.
* **Проект — Слой** – отношение, определяющее перечень слоёв проекта. Данное отношение имеет тип один ко многим, так как один проект может содержать несколько слоёв.
* **Слой — Объект** – отношение, обозначающее перечень объектов, содержащихся на слое. Данное отношение также является отношением многие ко многим, так как несколько объектов могут содержаться на слое и один объект может быть включен в разные слои.

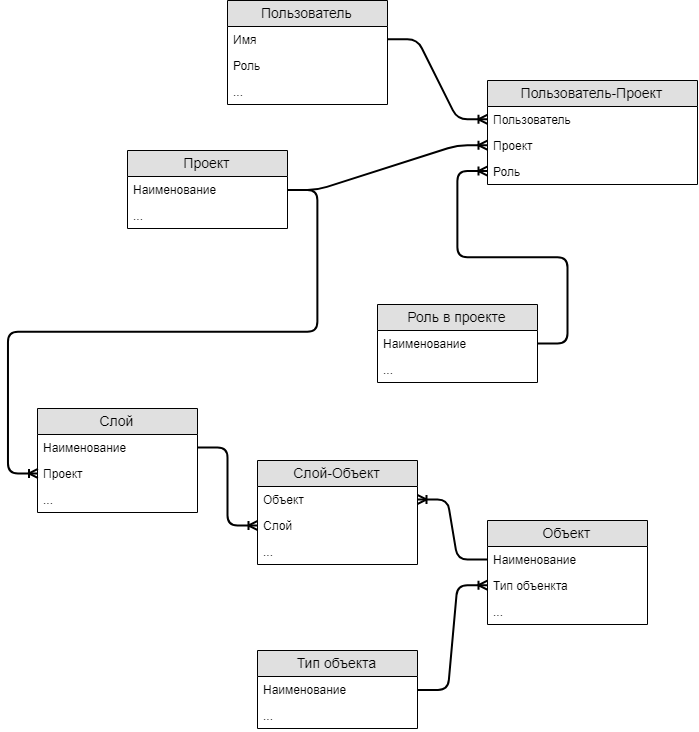


Рис. – ER-Диаграмма

На основании спроектированной ER-диаграммы составим структуру таблиц базы данных.

**Role** – Роли пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | smallint unsigned | not null | Идентификатор роли |
| name | char (100) | default “” | Наименование роли |
| primary key(id) | | | Первичный ключ |
| unique key(name) | | | Уникальный ключ |

**User** – Пользователи

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор пользователя |
| name | char (100) | default “” | Имя пользователя |
| login | char (100) | default “” | Логин |
| pass | char (100) | default “” | Пароль |
| role\_id | smallint unsigned |  | Идентификатор роли |
| email | char (100) | default “” | Электронная почта |
| primary key(id) | | | Первичный ключ |
| unique key(login) | | | Уникальный ключ |
| key(role\_id) | | |  |
| key(email) | | |  |
| foreign key role\_id references role(id) on delete restrict | | | Внешний ключ |

**Project** – Проекты

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор проекта |
| name | char (150) | default “” | Наименование проекта |
| autor | char (150) | default “” | Автор проекта |
| date\_start | datetime | default “” | Дата начала проекта |
| date\_edit | datetime |  | Дата и время последней модификации |
| scope | char (150) | default “” | Область применения (сфера деятельности) |
| primary key(id) | | | Первичный ключ |
| unique key(name) | | | Уникальный ключ |
| key(date\_start) | | |  |
| key(date\_edit) | | |  |
| key(scope) | | |  |

**Project\_user** – Проекты-пользователи

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор записи |
| project\_id | int unsigned | not null | Идентификатор проекта |
| user\_id | int unsigned | not null | Идентификатор пользователя |
| role\_id | smallint unsigned | not null | Идентификатор роли пользователя в проекте |
| primary key(id) | | | Первичный ключ |
| unique key (project\_id, user\_id) | | | Уникальный ключ |
| key(project\_id) | | |  |
| key(user\_id) | | |  |
| key(role\_id) | | |  |
| foreign key project\_id references project(id) on delete restrict | | | Внешний ключ (Проекты) |
| foreign key user\_id references user(id) on delete restrict | | | Внешний ключ (Пользователи) |
| foreign key role\_id references role(id) on delete restrict | | | Внешний ключ |

**Layer** – Слои

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор слоя |
| name | char (100) | default “” | Имя пользователя |
| project\_id | int unsigned | not null | Идентификатор проекта |
| activity | enum(‘0’, ‘1’) | Default ‘0’ | Активность – отображение/сокрытие слоя |
| primary key(id) | | | Первичный ключ |
| unique key(name) | | | Уникальный ключ |
| key(project\_id) | | |  |
| key(activity) | | |  |
| foreign key project\_id references project(id) on delete restrict | | | Внешний ключ (Проекты) |

**Type\_obj** – Типы объектов

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор типа |
| name | char (100) | default “” | Наименование типа |
| primary key(id) | | | Первичный ключ |
| unique key(name) | | | Уникальный ключ |

**Object** – Объект

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор объекта |
| name | char (100) | default “” | Наименование объекта |
| layer\_id | int unsigned | not null | Идентификатор слоя |
| type\_id | mediumint unsigned | not null | Идентификатор типа объекта |
| coords | mediumtext |  | Координаты объекта – объект GeoJSON |
| geometry | char (50) | default ‘’ | тип графического примитива, представляющий его на карте |
| icon | char (150) | default ‘’ | Иконка ссылка на файл с иконкой |
| length | decimal (12,4) | default 0 | максимальная длина объекта |
| width | decimal (12,4) | default 0 | максимальная ширина объекта |
| height | decimal (12,4) | default 0 | максимальная высота объекта |
| area | decimal (12,4) | default 0 | площадь объекта |
| volume | decimal (12,4) | default 0 | объем объекта |
| primary key(id) | | | Первичный ключ |
| unique key (name, layer\_id) | | | Уникальный ключ |
| key(type\_id) | | |  |
| key(geometry) | | |  |
| foreign key layer\_id references layer(id) on delete restrict | | | Внешний ключ (Слои) |
| foreign key type\_id references type\_obj(id) on delete restrict | | | Внешний ключ (type\_obj) |

**Layer\_object** – Слои-объекты

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Дополнительно** | **Описание** |
| id | int unsigned | not null | Идентификатор записи |
| object\_id | int unsigned | not null | Идентификатор объекта |
| layer\_id | int unsigned | not null | Идентификатор слоя |
|  | | | |
| primary key(id) | | | Первичный ключ |
| unique key(name) | | | Уникальный ключ |
| key(layer\_id) | | |  |
| key(object\_id) | | |  |
| foreign key layer\_id references layer(id) on delete restrict | | | Внешний ключ (Слои) |
| foreign key object\_id references object(id) on delete restrict | | | Внешний ключ (Объекты) |

**Глава 3. Реализация геоинформационной системы**

Для решения поставленной задачи, в качестве среды разработки, была выбрана PhpStorm 2022.

PhpStorm – это интегрированная среда разработки на PHP с интеллектуальным редактором, которая глубоко понимает код, поддерживает PHP 5.3-8.1, обеспечивает умное автодополнение кода, рефакторинги, предотвращение ошибок на лету и поддерживает смешивание языков.

Среди ключевых особенностей PhpStorm можно выделить:

* Интеллектуальный редактор PHP кода с подсветкой синтаксиса, автодополнением кода, расширенными настройками форматирования кода, предотвращением ошибок налету, детектор дублируемого кода;
* Поддерживает PHP 5.3-8.1, генераторы, сопрограммы и все синтаксические улучшения;
* Поддержка Docker, Composer, встроенный REST клиент, Command Line Tools, SSH консоль;
* Поддержка фреймворков (MVC view для Symfony2, Yii) и специализированные плагины для ведущих PHP фреймворков (Laravel, Symfony, Magento, Drupal, Yii, CakePHP, WordPress, Joomla! и др.);
* HTML, CSS, JavaScript редактор. Отладка и модульное тестирование для JS. Поддержка передовых технологий веб-разработки, включая HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, JavaScript, TypeScript, ECMAScript Harmony, Emmet и др.;
* Полный набор инструментов для фронтенд-разработки;
* Интеграция с системами управления версиями (Git, SVM, CVC и др.), включая унифицированный интерфейс;
* Удаленное развертывание приложений и автоматическая синхронизация с использованием FTP, SFTP, FTPS и др.;
* Интеграция с баг-трекерами;
* Инструменты работы с базами данных, SQL редактор;
* Кроссплатформенность (Windows, Mac OS X, Linux).

# **Разработка функции сохранения пространственных объектов**

В качестве промежуточного формата хранения пространственных объектов было принято решение использовать JSON-формат.

**JSON** (англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).

JSON-текст представляет собой (в закодированном виде) одну из двух структур [1]:

* Набор пар ключ-значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка, значением – любая форма. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты – учитывать только первый такой ключ, учитывать только последний такой ключ, генерировать ошибку.
* Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

Структуры данных, используемые JSON, поддерживаются любым современным языком программирования, что и позволяет применять JSON для обмена данными между различными языками программирования и программными системами.

В качестве значений в JSON могут быть использованы:

* запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
* массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, то есть не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.
* число (целое или вещественное).
* литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
* строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

Следующий пример показывает JSON-представление данных об объекте, описывающем человека. В данных присутствуют строковые поля имени и фамилии, информация об адресе и массив, содержащий список телефонов. Как видно из примера, значение может представлять собой вложенную структуру.



Рис. 3 – Пример структуры JSON

В качестве значений в JSON могут быть использованы как числа, так и строки. Поэтому запись "postalCode": "101101" содержит строку, а "postalCode": 101101 – уже числовое значение. Из-за слабой типизации в JavaScript и PHP строка может быть приведена к числу и не влиять на логику программы. Тем не менее, рекомендуется аккуратно обращаться с типом значения, так как JSON служит для межсистемного обмена.

**Экспортирование объектов на карте в JSON-формат**

При создании объекта необходимо задать ему название. Помимо названия можно задавать любые свойства объекта в неограниченном количестве. Причём идентификационный номер каждому объекту присваивается автоматически. Пример всплывающего окна с формой записи названия объекта представлен на рисунке ниже.

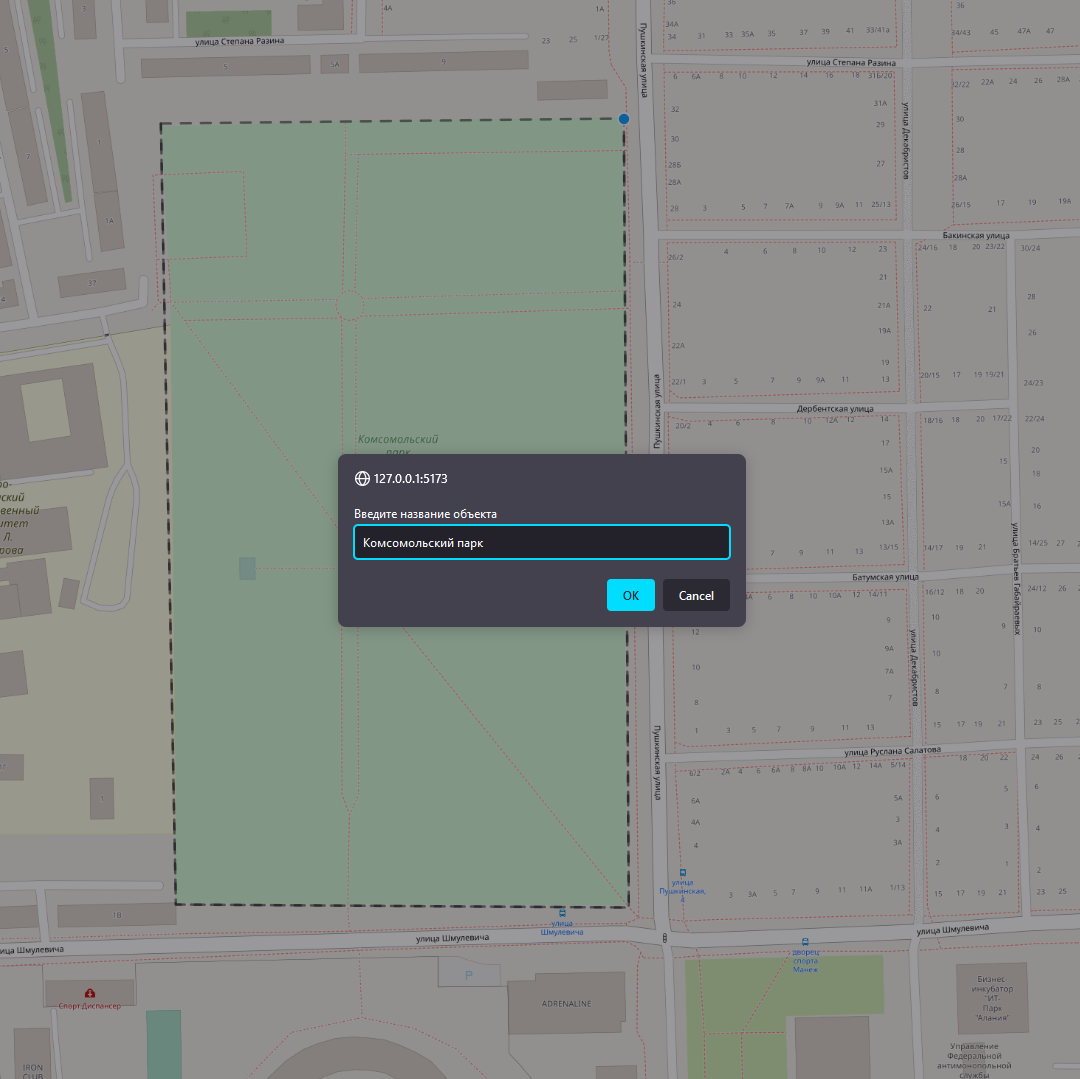


Рис. 4 – Окно с полем записи названия объекта

В результате экспорта был получен JSON-файл, в структуру которого входят следующие строки:

* type – описывает тип исходного геометрического объекта;
* coordinates – координаты вершин объекта. В данном случае это полигон с четырьмя вершина;
* properties – задаваемые заранее или при создании свойства объекта, среди которых:
  + id – идентификационный номер объекта;
  + name – название объекта, которое задаётся при создании;
  + description – описание объекта (например, состояние, площадь и т.д.).

Помимо временного хранения объектов в формате JSON для последующего сохранения их в таблице БД, в программе предусмотрена функция сохранения JSON-файла на локальное хранилище. Это может быть полезно для дальнего использования этих объектов в других ГИС.

Добавлена функция сохранения выбранной части карты в формате PDF для последующей печати. Функция поддерживает формат серии «A» и размеры от 0 до 5.

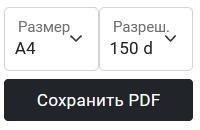


Рис. 5 – Интерфейс для сохранения PDF

Также была реализована функция подсчета и отображения длин линий и сторон геометрических полигонов. Программа ко всему прочему может вычислять площадь построенного полигона. При желании данную функцию можно отключить. Пример отображения площади полигона можно увидеть на изображении ниже.



Рис. 6 – Отображение площади и длин сторон

# **Реализация функции растровых и векторных слоёв в ГИС**

В настоящее время использование геоинформационных систем (ГИС) становится все более распространенным. Однако, с ростом количества данных и сложности задач возникает необходимость улучшения функциональности ГИС. В связи с этим, было принято решение о внедрении системы слоев в ГИС. Система слоев представляет собой мощный инструмент для управления отображением данных на карте. В данном отчете будет описан процесс внедрения системы слоев в ГИС и ожидаемые результаты от этого процесса.

Слои в геоинформационной системе – это механизм, используемый для отображения географических наборов данных в ГИС. Каждый слой ссылается на набор данных и определяет, как этот набор будет показан при помощи символов и текстовых меток. Слои – это картографические данные, хранящиеся на диске компьютера в специальных ГИС-форматах. Любые географические объекты, расположенные на картах один поверх другого, могут быть разделены на разные слои.

Система слоев в геоинформационной системе предоставляет следующие преимущества:

* Удобное управление и отображение данных на карте;
* Возможность настройки отображения данных на карте;
* Изменение стилей и символов для каждого слоя;
* Настройка порядка отображения слоев;
* Возможность изменения прозрачности каждого слоя.

Например, система слоев позволяет удобно управлять и отображать данные на карте, а также настраивать порядок отображения слоев и прозрачность каждого слоя. Это позволяет пользователям геоинформационной системы быстро и эффективно работать с данными и получать необходимую информацию.

Помимо всего прочего система слоёв предоставляет следующие преимущества:

* Возможность настройки фильтрации данных;
* Возможность настройки масштабирования данных;
* Возможность настройки взаимодействия между слоями;
* Возможность настройки доступа к данным для разных пользователей.

Система слоев позволяет настраивать фильтрацию данных и масштабирование данных, что позволяет пользователям геоинформационной системы получать необходимую информацию в удобном формате. Кроме того, система слоев позволяет настраивать взаимодействие между слоями и доступ к данным для разных пользователей, что повышает эффективность работы с данными.

Цифровая карта может быть организована в виде множества слоев (покрытий или карт подложек). Слои в ГИС представляют набор цифровых картографических моделей, построенных на основе объединения (типизации) пространственных объектов, имеющих общие функциональные признаки. Совокупность слоев образует интегрированную основу графической части ГИС.

**Растровые слои**

Растровые данные используются в ГИС когда необходимо отобразить непрерывное по площади явление, которое нельзя легко разбить на векторные объекты. Точечные, линейные и полигональные объекты хорошо подходят для отображения некоторых объектов ландшафта, например деревьев, дорог и зданий. Другие объекты отобразить при помощи векторных объектов сложнее. Например, поля состоят из множества участков с разным цветом и плотностью покрытия. Можно было бы создать по одному полигону на каждое поле, но так мы потеряем большую часть информации из-за упрощения всех объектов в один полигон.

Это происходит из-за того, что атрибуты векторного объекта применяются ко всему объекту, именно по этой причине вектор не лучший выбор для отображения разнородных (не идентичных) объектов. Другим решением была бы оцифровка каждого небольшого участка, отличающегося цветом травы и покрытием. Недостаток такого подхода в том, что потребуется очень много времени и сил для создания хорошего набора векторных данных.

Решением этих проблем является использование растровых данных. Во многих случаях используются растровые данные в качестве подложки под векторные слои, чтобы улучшить восприятие содержащейся в них информации. Человеческий глаз очень хорошо распознает образы, поэтому использование изображения под векторными данными делает карты более понятными и удобочитаемыми.

Растровые данные хорошо подходят не только для изображений реальной поверхности (например, спутниковые изображения или аэрофотосъемка), но и для отображения абстрактной информации. К примеру, растр может использоваться для визуализации тенденции осадков на протяжении года, или для отображения вероятности пожара. В таких случаях каждая ячейка растра содержит некоторую величину, например вероятность возникновения пожара по десятибалльной шкале.

**Векторные слои**

Векторный слой – это слой, содержащий группы похожих объектов и их связанные свойства. Векторные слои – это метод представления классов пространственных объектов. Объекты в векторном слое могут быть точками, линиями или полигонами. Векторные слои лучше всего использовать для визуализации данных поверх базовых растровых карт.

В векторном слое графические объекты независимо от их графического типа делятся на две разновидности: простые графические объекты (примитивы) и типовые (классифицированные) графические объекты.

Простые графические объекты содержат все атрибуты отображения внутри себя. Типовые графические объекты содержат лишь ссылку на типовую структуру, которая и определяет графический тип, атрибуты отображения и текущее состояние объекта (такие объекты, как правило, используют при нанесении инженерных сетей).

Простые графические объекты могут быть связаны с одной семантической базой данных, общей для всего слоя. Типовые графические объекты связываются только с семантической базой своего типа.

Любой слой может иметь свою библиотеку символов для отображения точечных объектов. Символ представляет собой группу графических примитивов (линий, полигонов, окружностей, текста), имеющих свой стиль, цвет и т.д. Каждая такая группа имеет точку привязки и угол поворота всей группы вокруг этой точки. Кроме того, символ может иметь пользовательское название.

Каждый векторный слой имеет библиотеку стилей заливок для площадных объектов и стилей для линейных объектов.

Все векторные слои могут иметь собственную библиотеку типов объектов. Каждый тип описывает площадной, линейный или символьный типовой графический объект, имеет пользовательское название и может быть связан с собственной семантической базой данных.

Типовой графический объект принадлежит к одному из типов в библиотеке типовых объектов векторного слоя и находится в одном из режимов данного типа. Отображение объекта зависит от того, как отображается текущий режим объекта данного типа.

Типовые объекты по графическому виду могут делиться на:

* Символьные – они отображаются с помощью символа, определяется на местности координатой точки привязки и углом поворота символа вокруг точки привязки. Каждый режим такого типа обязательно связан с одним из символов библиотеки символов. Для решения различных задач, связанных с инженерными сетями, символьный объект может иметь дополнительный признак, конкретизирующий назначение типа: источник, потребитель, отсекающее устройство или просто узел.
* Линейные – представляет собой ломаную. Каждый режим линейного объекта имеет свой цвет, толщину и стиль. Типовой линейный объект может обладать признаком того, что данный тип является участком. Отличие участка от простой ломаной состоит в том, что начало и конец такой ломаной обязательно должны быть связаны с типовыми символьными объектами, т.е. начинаться символьным объектом и заканчиваться символьным объектом.
* Площадные – представляют собой замкнутый контур. Каждый режим объекта имеет свой цвет, толщину и стиль линии контура, а также цвет и стиль заливки внутренней области контура.
* Текстовые – представляют собой текст с заданными параметрами, такими как шрифт, начертание, размер, выравнивание.

**Создание системы слоёв при помощи OpenLayers**

Для создания системы слоёв при помощи OpenLayers необходимо:

* Подключить библиотеку OpenLayers к HTML-документу;
* Создать объект карты с указанием элемента контейнера и параметров отображения;
* Добавить нужные слои к объекту карты с помощью метода addLayer или свойства layers;
* Настроить свойства и стили каждого слоя в соответствии с потребностями ГИС.

Ниже представлен стандартный пример кода, который создаёт карту с двумя слоями: базовым слоем OSM (Open Street Map) и векторным слоем с точками:

var map = new ol.Map({

target: 'map', // id элемента контейнера

view: new ol.View({

center: ol.proj.fromLonLat([37.6173, 55.7558]), // центр карты в координатах EPSG:3857

zoom: 10 // уровень масштабирования

})

});

// Создаем базовый слой OSM

var osmLayer = new ol.layer.Tile({

source: new ol.source.OSM() // источник тайлов OSM

});

// Добавляем базовый слой к карте

map.addLayer(osmLayer);

var vectorLayer = new ol.layer.Vector({

source: new ol.source.Vector({ // источник векторных данных

features: new ol.format.GeoJSON().readFeatures(points) // читаем данные из GeoJSON

}),

});

// Добавляем векторный слой к карте

map.addLayer(vectorLayer);

Однако данный способ не подходит для поставленной задачи, поскольку не позволяет добавлять, удалять и переключаться между векторными слоями. Поэтому было принято решение написать собственный алгоритм.

Для начала необходимо написать функцию добавления слоёв на карту. Код функции добавления слоя на карту представлен ниже:

var selectID = document.getElementById('vector\_layers');

// Функция отслеживания нажатия на кнопку создания слоя

$(".create").click(function() {

    let newSelectOption = new Option('Layer ' + (selectID.length - 1), selectID.length - 1);

// Добавляем слой в меню выбора

    selectID.options[selectID.length] = newSelectOption;

});

Добавленные векторные слои необходимо хранить где-то в памяти для последующего переключения между ними. Был разработан следующий алгоритм:

// Массив со значениями опций выбора слоя

var values = new Array();

$("#vector\_layers option").each(function() {

    if($(this).val() !== '') {

        values.push($(this).val())

    }

});

// Массив для хранения источников векторных слоёв

var sources = new Array();

// Массив для хранения векторных слоёв

var vectors = new Array();

// Функция отслеживания изменения текущего слоя

document.getElementById('vector\_layers').addEventListener('change', function() {

// Цикл для записи всех созданных слоёв на карте в массив

    for (let index = 0; index < values.length-1; index++) {

        var s = new VectorSource();

        var v = new VectorLayer({

            source: s

        })

        sources.push(s);

        vectors.push(v);

    }

});

Далее необходимо было написать функцию переключения слоёв на карте, при этом слои должны сохранять размещенные на себе объекты для последующего применения в проекте. Ниже представлен код этой функции:

// Функция отслеживания изменения текущего слоя

document.getElementById('vector\_layers').addEventListener('change', function() {

    if (this.value !== 'layer') {

        for (let index = 0; index < sources.length; index++) {

            if (index != this.value) {

// Удаляем предыдущий слой с карты

                map.removeLayer(vectors[index]);

                removeInteraction(sources[index]);

            }

        }

// Добавляем текущий слой на карту

        addInteraction(sources[this.value]);

        map.addLayer(vectors[this.value]);

    }

});

Помимо функции добавления слоя на карту, необходимо также реализовать функционал удаления ненужных слоёв. Код реализации данной функции представлен ниже:

$(document).ready(function() {

// Функция отслеживания нажатия на кнопку удаления слоя

    $('#delete\_layer').click(function() {

        // Удаляем выбранный слой из списка слоёв

        $('#vector\_layers option:selected').remove();

    });

});

Код для отображения основного слоя с картой:

const raster = new TileLayer({

    source: new OSM(),

});

const map = new Map({

    layers: [

        raster,

    ],

    target: 'map',

    view: new View({

        center: fromLonLat([0, 0]),

        zoom: 1,

     }),

});

Функция для размещения геометрических примитивов на карте:

const typeSelect = document.getElementById('geom\_type');

var draw, modify; // global so we can remove it later

function addInteraction(vector) {

    const value = typeSelect.value;

    if (value !== 'None') {

        draw = new Draw({

            source: vector,

            type: typeSelect.value,

        });

        map.addInteraction(draw);

        modify = new Modify({

            source: vector,

        });

        map.addInteraction(modify);

    }

    typeSelect.onchange = function () {

        map.removeInteraction(draw);

        addInteraction(vector);

    };

}

function removeInteraction() {

    map.removeInteraction(draw);

}

Демонстрация графического интерфейса программы представлен ниже на рис. 1 и 2. Как видно из рисунка 1, программа позволяет размещать геометрические примитивы на векторном слое, при этом слоёв может быть неограниченное количество. Пример размещения объектов на созданном слое можно увидеть на рисунке ниже.

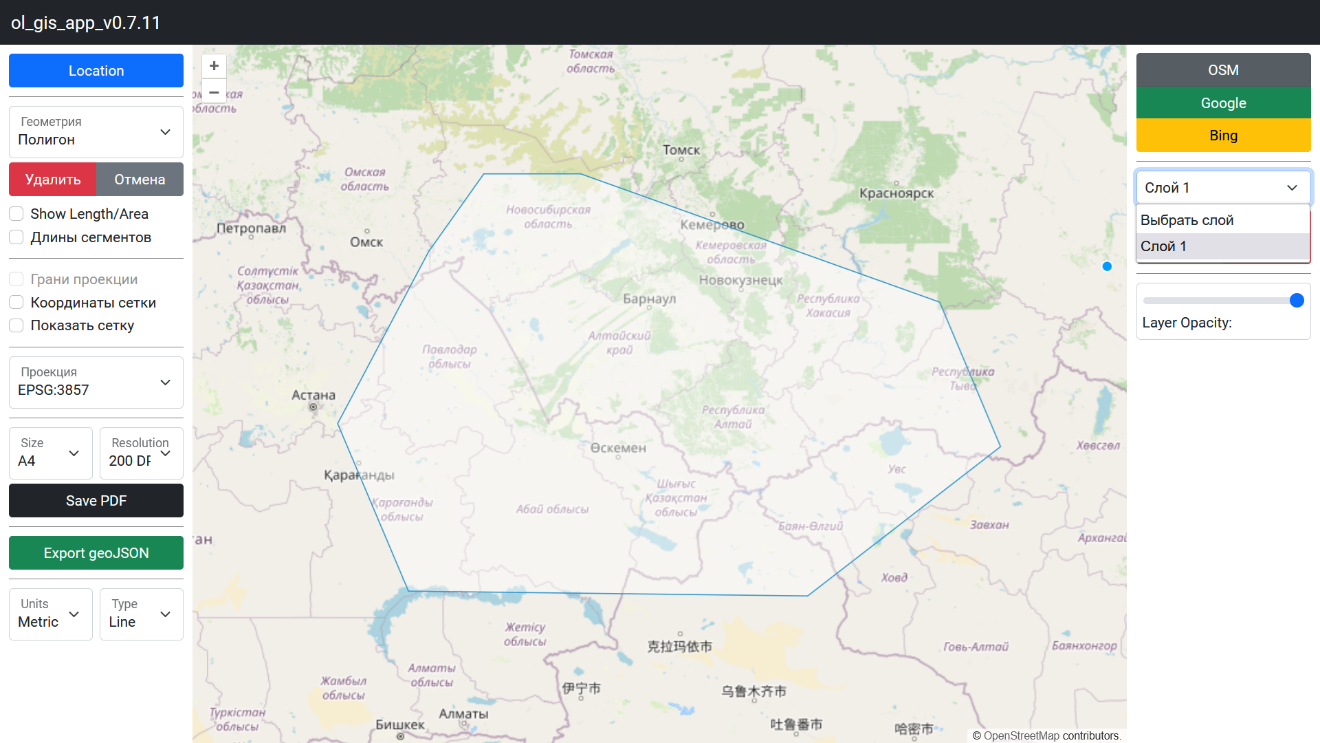


Рис. 1. Демонстрация слоя №1

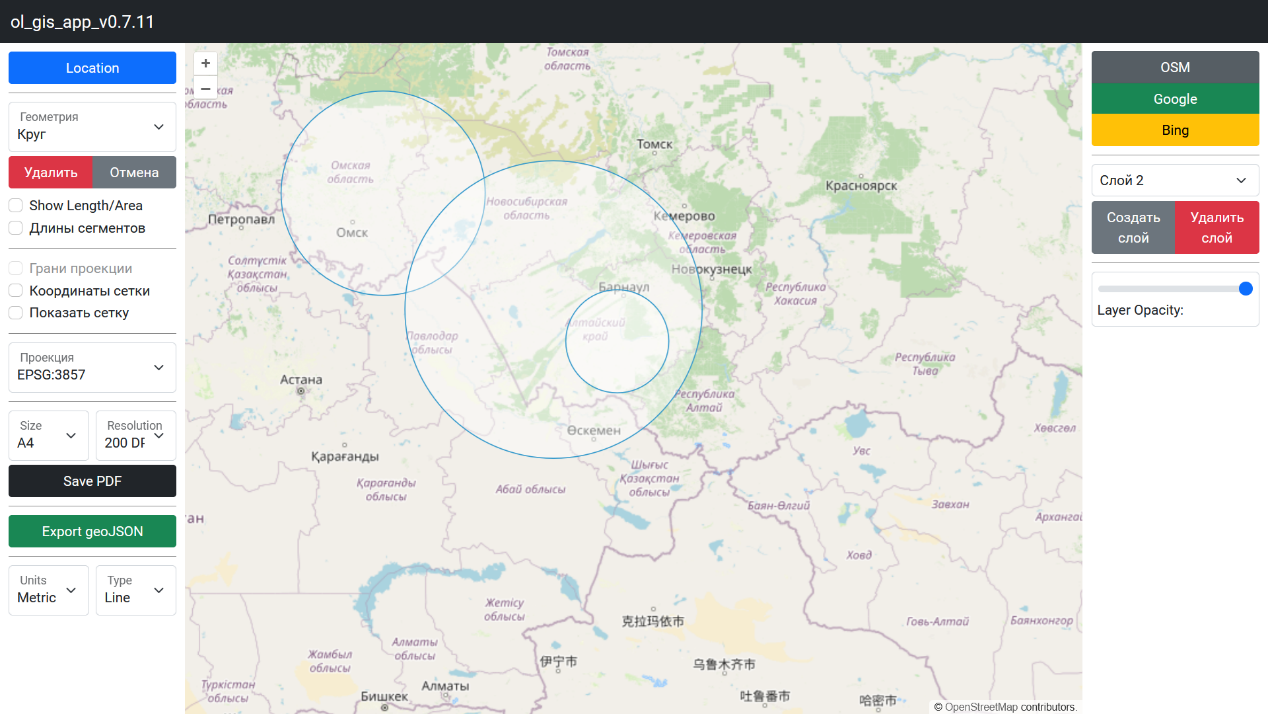


Рис. 2. Демонстрация слоя №2

Для создания нового слоя необходимо нажать на кнопку «Создать слой», для удаления текущего слоя – «Удалить слой».

**Заметка #1**

В настоящее время использование геоинформационных систем (ГИС) становится все более распространенным. Однако, с ростом количества данных и сложности задач возникает необходимость улучшения функциональности ГИС. В связи с этим, было принято решение о внедрении системы слоев в ГИС. Система слоев представляет собой мощный инструмент для управления отображением данных на карте. В данном отчете будет описан процесс внедрения системы слоев в ГИС и ожидаемые результаты от этого процесса.  
  
Слои в геоинформационной системе – это механизм, используемый для отображения географических наборов данных в ГИС. Каждый слой ссылается на набор данных и определяет, как этот набор будет показан при помощи символов и текстовых меток. Слои – это картографические данные, хранящиеся на диске компьютера в специальных ГИС-форматах. Любые географические объекты, расположенные на картах один поверх другого, могут быть разделены на разные слои.  
  
Система слоев в геоинформационной системе предоставляет следующие преимущества:  
·        Удобное управление и отображение данных на карте;  
·        Возможность настройки отображения данных на карте;  
·        Изменение стилей и символов для каждого слоя;  
·        Настройка порядка отображения слоев;  
·        Возможность изменения прозрачности каждого слоя.  
  
Например, система слоев позволяет удобно управлять и отображать данные на карте, а также настраивать порядок отображения слоев и прозрачность каждого слоя. Это позволяет пользователям геоинформационной системы быстро и эффективно работать с данными и получать необходимую информацию.  
  
Помимо всего прочего система слоёв предоставляет следующие преимущества:  
·        Возможность настройки фильтрации данных;  
·        Возможность настройки масштабирования данных;  
·        Возможность настройки взаимодействия между слоями;  
·        Возможность настройки доступа к данным для разных пользователей.  
  
Система слоев позволяет настраивать фильтрацию данных и масштабирование данных, что позволяет пользователям геоинформационной системы получать необходимую информацию в удобном формате. Кроме того, система слоев позволяет настраивать взаимодействие между слоями и доступ к данным для разных пользователей, что повышает эффективность работы с данными.  
Цифровая карта может быть организована в виде множества слоев (покрытий или карт подложек). Слои в ГИС представляют набор цифровых картографических моделей, построенных на основе объединения (типизации) пространственных объектов, имеющих общие функциональные признаки. Совокупность слоев образует интегрированную основу графической части ГИС.

**Уникальность: 77.12%**

**Заметка #2 (Введение)**

Геоинформационные системы (ГИС) – это системы сбора, хранения, обработки, доступа, анализа, интерпретации и графической визуализации пространственных данных. ГИС широко применяются в различных сферах деятельности, таких как картография, геология, экология, управление территорией, транспорт и многие другие. ГИС позволяют решать сложные задачи пространственного моделирования, мониторинга, планирования и поддержки принятия решений на основе актуальной и достоверной информации о географических объектах и явлениях.

ГИС основаны на использовании *данных дистанционного зондирования Земли* (ДЗЗ), которые представляют собой изображения земной поверхности, полученные с помощью специальных аппаратов на космических или воздушных платформах. Данные ДЗЗ имеют высокое пространственное, временное и спектральное разрешение, что позволяет детально и оперативно исследовать состояние окружающей среды и использовать природные ресурсы.

Однако для эффективного использования ГИС необходимо иметь удобный и надежный инструмент для создания, редактирования, публикации и распространения геоинформационных продуктов. Таким инструментом может быть веб-платформа для создания геоинформационных систем.

Веб-платформа – это программное обеспечение, которое работает в сети Интернет и предоставляет пользователям возможность создавать свои собственные веб-сайты или приложения с помощью стандартных технологий и инструментов.

Веб-платформа для создания геоинформационных систем – это веб-платформа, которая специализируется на работе с геоданными и предоставляет пользователям функционал для создания веб-карт, веб-ГИС, веб-сервисов и других геоинформационных продуктов.

Целью данной работы является разработка веб-платформы для создания геоинформационных систем на основе современных технологий и стандартов. Для достижения этой цели необходимо решить следующие задачи:

* провести обзор существующих веб-платформ для создания геоинформационных систем и выделить их основные характеристики, преимущества и недостатки;
* выбрать подходящие технологии и инструменты для разработки веб-платформы (например, языки программирования, фреймворки, библиотеки и т.д.);
* разработать прототип веб-платформы для создания геоинформационных систем с целью тестирования и последующего добавления необходимого функционала;
* ???

Уникальность: 78.51%

**Заметка #3**

Методика разработки веб-платформы для создания геоинформационных систем может состоять из следующих этапов:

Анализ требований к веб-платформе. На этом этапе необходимо определить целевую аудиторию, функциональные и нефункциональные требования, архитектуру и дизайн веб-платформы, а также выбрать подходящие технологии и инструменты для разработки.

Проектирование веб-платформы. На этом этапе необходимо разработать детальный план реализации веб-платформы, включая структуру базы данных, интерфейс пользователя, модули и компоненты, алгоритмы и методы обработки геоданных, тестирование и документирование.

Разработка веб-платформы. На этом этапе необходимо реализовать веб-платформу согласно проектному плану, используя выбранные технологии и инструменты. На этом этапе также необходимо проводить регулярное тестирование и отладку кода, а также контроль качества и безопасности веб-платформы.

Развертывание и поддержка веб-платформы. На этом этапе необходимо развернуть веб-платформу на выбранном хостинге или сервере, обеспечить ее доступность и работоспособность для пользователей, а также проводить мониторинг, обновление и сопровождение веб-платформы.

Для разработки веб-платформы для создания геоинформационных систем можно использовать различные методологии разработки программного обеспечения, такие как каскадная, спиральная, прототипирование или гибкие (agile) методологии. Выбор методологии зависит от многих факторов, таких как размер и сложность проекта, требования заказчика, ресурсы и сроки разработки. В целом, гибкие методологии предпочтительнее для разработки веб-платформы для создания геоинформационных систем, так как они позволяют быстрее и лучше адаптироваться к изменяющимся требованиям и потребностям пользователей.

**Заметка #4 (OpenLayers)**

OpenLayers – это JavaScript библиотека с открытым исходным кодом, рассчитанная для реализации и поддержки карт на основе API (программного интерфейса). В OpenLayers входят следующие компоненты: JavaScript Rico и PJF (Prototype JavaScript Framework).

OpenLayers предоставляет возможность разработчику легко разместить на любой веб-странице динамическую карту, на которой будут отдельные слои (округов, городов, улиц, дорог и т.д.), размещённые на разных серверах, например, GeoServer, ArcGIS, Mapserver и т.д. Она может отображать плитки карты, векторные данные и маркеры из любого источника. Благодаря OpenLayers можно с быстротой и лёгкостью разработать веб-интерфейс, необходимый для отображения картографических материалов. Эти материалы могут быть представлены в разных форматах и расположены на различных серверах.

Помимо всего вышеперечисленного, OpenLayers обладает и другими интересными особенностями и возможностями:

* Размещение навигационной панели;
* Перемещение по карте при помощи мыши;
* Изменение масштаба карты;
* Получение координат точки по щелчку мыши;
* Использование функции отключения слоев карты;
* Получение информации о выбранных объектах;
* Изменение прозрачности слоя;
* Размещение на карте геометрических элементов (точки, линии или полигона).

**Уникальность: 52.69%**

Поскольку OpenLayers представляет собой библиотеку, написанную на JavaScript, то отпадает необходимость установки в привычном смысле этого слова, для работы с OpenLayers достаточно скачать библиотеку и распаковать её в каталог проекта.

При работе с OpenLayers API необходимо учитывать два основных момента: объект Map и объект Layer. Map позволяет хранить основную информацию о параметрах карты, например, информацию о проекции, единицах измерения и т.д. Данные устанавливаются с помощью объектов Layer. Объект Layer хранит в себе информацию о слоях данных, помещенных на карту.

Пример создания объекта Map для отображения карты. Для начала необходимо создать файл «index.html», который будет использоваться для встраивания объекта OpenLayers, отвечающий за отображение карт.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="node\_modules/ol/ol.css">

<style>

.ol-map33 {

width: 666px; height: 666px;

}

</style>

</head>

<body>

<div id="opnlrs\_map13" class="ol-map33"></div>

<script src="https://cdn.jsdelivr.net/npm/elm-pep@1.0.6/dist/elm-pep.js"></script>

<script type="module" src="main.js"></script>

</body>

</html>

Далее необходимо создать файл «main.js» с объектом opnLrs\_map13, который отвечает за отображение карты.

import Map from 'ol/Map.js';

import OSM from 'ol/source/OSM.js';

import TileLayer from 'ol/layer/Tile.js';

import View from 'ol/View.js';

var opnLrs\_map13 = new Map({

layers: [

new TileLayer({

source: new OSM(),

}),

],

target: 'opnlrs\_map13',

view: new View({

center: [4193730, 7505925],

zoom: 3,

}),

});

**Уникальность: 53.24%**

**Заметка #5 (Введение)**

Географические информационные системы (ГИС) – это инструменты и технологии, используемые для сбора, хранения, управления, анализа и отображения географически привязанных данных. ГИС позволяют пользователям визуализировать, анализировать, хранить и интерпретировать цифровые данные различными способами, что делает их бесценными инструментами для анализа пространственных и географических данных.

ГИС используются во многих областях, включая бизнес, здравоохранение, общественную безопасность и транспорт, и предлагают целый ряд возможностей, которые помогают пользователям лучше понять географию определенного региона, выявить закономерности и принимать более обоснованные решения.

Геоинформационные системы (ГИС) - это системы сбора, хранения, анализа и графической визуализации пространственных (географических) данных и связанной с ними информации о необходимых объектах. ГИС широко применяются в различных областях науки, техники, экономики, экологии и управления, так как позволяют решать сложные задачи пространственного моделирования, мониторинга, планирования и поддержки принятия решений. ГИС основаны на использовании данных дистанционного зондирования Земли (ДЗЗ), которые представляют собой изображения земной поверхности, полученные с помощью специальных аппаратов на космических или воздушных платформах.

Данные ДЗЗ имеют высокое пространственное, временное и спектральное разрешение, что позволяет детально и оперативно исследовать состояние окружающей среды и использовать природные ресурсы.

Целью данной работы является изучение основных принципов и методов построения и использования геоинформационных систем с целью дальнейшей разработки веб-платформы для создания геоинформационных систем.

* провести обзор существующих ГИС (например, ArcGIS, QGIS, gvSIG и т.д.) и их классификацию по различным критериям;
* разработать структуру базы данных ГИС и способы ее заполнения;
* спроектировать и разработать подходящий программный продукт для создания ГИС;
* определить источники и форматы данных ДЗЗ для ГИС;
* оценить достоинства и недостатки использования ГИС для решения поставленной задачи.

Предполагается, что работа будет выполнена с использованием современных компьютерных технологий и интернет-ресурсов. Работа будет состоять из введения, трех глав, заключения и списка литературы. В первой главе будет дан теоретический обзор ГИС и ДЗЗ, во второй - описан процесс создания ГИС для выбранной предметной области, в третьей - представлены результаты анализа данных ДЗЗ с помощью ГИС. В заключении будут подведены итоги работы и сделаны выводы по ее результатам.

**Заметка #6 – JS**

JavaScript – язык программирования, поддерживающий несколько парадигм программирования: объектно-ориентированный, императивный и функциональный стили. В первую очередь применяется для добавления интерактивности в веб-страницах. JavaScript обладает большим количеством фреймворков (Angular, React, Node.js и т.д.), которые упрощают разработку приложений. К основным достоинствам можно отнести:

* Высокая скорость и производительность. JavaScript позволяет обрабатывать код на устройствах пользователей, что значительно уменьшает нагрузку на сервер и экономит Интернет-трафик;
* Отличная интеграция с веб-технологиями. Все популярные браузеры поддерживают скрипты, написанные на JavaScript. Интеграция с серверными частями приложений (backend);
* Простой в освоении синтаксис, низкий порог вхождения;
* Развитая инфраструктура. Для JS существует множество различных библиотек и фреймворков, например Angular, React, Node.js, Underscore и т.д.

**Уникальность: 83.23%**

**Заметка #7 – ER (Уникальность 75-86.51%)**

Инфологическая модель данных – это неформальное обобщенное описание разрабатываемой базы данных, при котором используются естественный язык, математические формулы, таблицы и графики, то есть вещи и концепции, наиболее понятные большинству людей. Основными элементами этой модели являются сущности, связи между ними и их свойства (атрибуты).

Концепция «сущность-связь» или ER-модель [7] (от англ. Entity-Relationship) – модель данных, позволяющая выделять главные сущности и связи между этими сущностями, при проектировании информационных систем с интегрированием баз данных, при отладке баз данных и исследовательской деятельности. Элементами ER-модели являются:

Сущность (Entity) – конкретный объект, который можно идентифицировать определённым образом, то есть отличающим его от других. Например, какой-то конкретный пользователь, сотрудник и так далее.

Связь (Relationship) – это соединение, благодаря которому устанавливается связь между двумя сущностями. Примером сущности может являться:

* Каждый клиент пользуется определённым количеством услуг. Две сущности – КЛИЕНТ и УСЛУГА – устанавливают между собой связь «использует». Другими словами, появляется связь КЛИЕНТ-УСЛУГА;
* Некой организацией руководит начальник, то сущности НАЧАЛЬНИК и ОРГАНИЗАЦИЯ устанавливают связь «управляет». Создаётся связь ОРГАНИЗАЦИЯ-РУКОВОДИТЕЛЬ;

Степень связи – показывает количество сущностей, участвующих в связи. Наиболее распространены следующие типы связей:

Связь «один к одному (1:1)» – вид связи, при которой единственному элементу из первой таблицы сопоставляется не более одного элемента во второй таблице и наоборот. Пример из реальной жизни: у одного человека есть одно единственное свидетельство о рождении, и наоборот – одно свидетельство о рождении является документом только у одного конкретного человека. То же касается и паспорта человека.

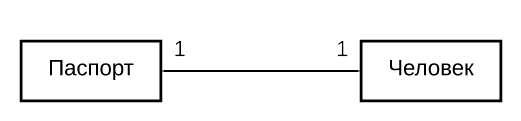


Рис. 4 – Связь «один к одному (1:1)»

Связь «один ко многим (1:M)» – вид связи, при которой одному элементу в первой таблице соответствует два или более элементов в другой таблице. Пример из реальной жизни: в одной школе может быть несколько классов, но каждый класс обязательно входит только в одну школу. То же касается и улицы с расположенными на ней домами.

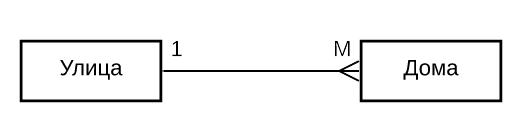


Рис. 5 – Связь «один ко многим (1:M)»

Связь «многие ко многим (M:M)» – вид связи, при которой множеству элементов одной таблицы соответствует множество элементов в другой таблице. Пример из реальной жизни: клиент может заказать несколько видов продукции, один вид продукции может быть куплен несколькими клиентами.

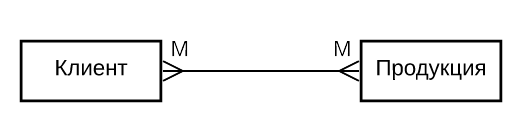
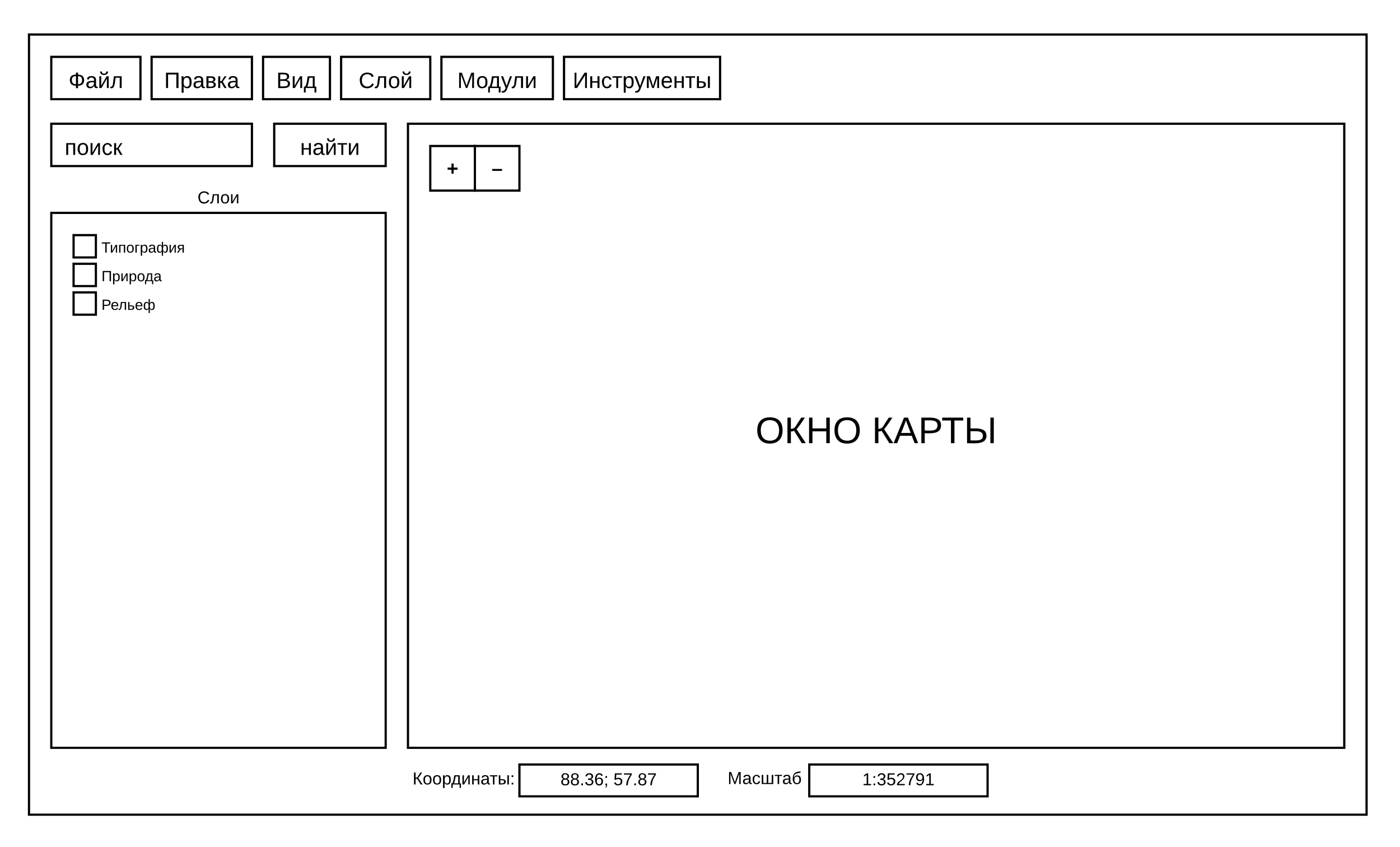


Рис. 6 – Связь «многие ко многим (M:M)»

**Макет интерфейса приложения**

Для взаимодействия с приложением был спроектирован эскиз интуитивно понятного и простого графического пользовательского интерфейса. Эскиз графического интерфейса окна взаимодействия с ГИС представлен на рисунке 1.

Рис. 1 – Макет интерфейса

**Заметка #8 – PHP Storm (Уникальность: 55.15%)**

Для разработки информационной системы была выбрана интегрированная среда разработки PhpStorm, созданная компанией JetBrains. PhpStorm – редактор для языков программирования PHP, HTML и JavaScript с интегрированной возможностью анализа кода в реальном времени, устранения ошибок в программном коде, а также автоматизированными средствами отладки кода для PHP и JavaScript. Данное программное решение является передовым на рынке разработки, позволяющее разработчикам проектировать и создавать приложения, удовлетворяющие всем современным требованиям.

Ключевые возможности PhpStorm:

* Встроенная среда разработки;
* Интегрированный редактор для HTML, CSS, JavaScript и PHP-кода;
* Подсветка синтаксиса. Наличие функции автоматического дополнения кода;
* Поддержка языка программирования PHP версии 5.3 и выше, плюс ко всему поддержка фреймворков PHP;
* Полный набор инструментов для вёрстки веб-страниц (front-end), то есть создания клиентской части сайта;
* Удобное интегрирование систем управления версиями;
* Продвинутые инструменты работы с БД, наличие SQL-редактора;
* Поддержка всех современных десктопных операционных систем.