1. **Введение**

Уважаемый председатель и члены комиссии. Вашему вниманию предлагается дипломная работа Лазарева Владимира, студента 4-ого курса, факультета ПММ на тему «Разработка новостного агрегатора KeepMePosted с клиентом Telegram».

На данный момент у человека есть множество различных источников новостей, такие как:

* интернет;
* ТВ, газеты, журналы (СМИ);
* Различные социальные сети, такие как ВКонтакте, Twitter.

В таком многообразии довольно непросто отслеживать только те темы, которые интересны человеку, смотреть новости только из тех источников, которым доверяет.

Учитывая это, целесообразно создать приложение, которое позволит своевременно получать обновления по указанным пользователем темам, источникам, и может предоставлять более обширную информацию.

1. **Постановка задачи**

Разработать веб-приложение *KeepMePosted*, реализующее систему для сбора новостей, *твитов* и прогноза погоды как через открытое API, так и с помощью анализа RSS-каналов. Клиенты программы должны быть представлены в виде *Telegram* бота и веб-страницы, предоставляющие следующие возможности:

* возможность просмотра самых свежих новостей;
* добавление и удаление категорий новостей, по которым пользователь будет получать обновления;
* возможность блокировать и снимать блокировку с новостных ресурсов;
* возможность просмотра самых новых твитов социальной сети Twitter;
* добавление и удаление отслеживаемых пользователей Twitter и хэштегов;
* возможность просмотра текущей погоды в указанных пользователем населенных пунктах;
* добавление городов в список отслеживаемых населенных пунктов;
* добавление и удаление городов для получения прогноза погоды;
* возможность просмотра новостей, твитов и погоды применяя пользовательские настройки, так и без них;
* включение и отключение оповещений, настройка их частоты отдельно для каждого из сервиса.

Для проектирования веб-интерфейса использовать библиотеку *Angular Material*.

1. **Анализ существующих решений**

Проанализировав несколько приложений и сервисов, представляющих собой новостные порталы и агрегаторы, была составлена таблица, представленная на слайде. В ней указаны преимущества и недостатки по многим критериям каждой программы.

Анализ показал, что они не эффективны для потребления информации, не удовлетворяют всем критериям. В связи с этим возникает задача создания системы, которая будет выполнять основные функции новостного агрегатора и обладать гибкими настройками.

1. **Интерфейс (главная страница Telegram-бота)**

Посмотрим на пользовательский интерфейс Telegram-бота. На данном слайде представлено главное состояние интерфейса бота, когда пользователь подключается к нему. Мы видим, что бот приветствует нас, и доступны 3 главных блока программы: новости, твиттер и погода.

1. **Интерфейс (главная страница сервисов Telegram-бота)**

При нажатии на любую из кнопок на главной странице мы попадаем в главное меню выбранного нами раздела. На данном этапе пользователю предоставляется выбор: вернуться обратно на главную страницу, перейти к просмотру информации или настроить сервис под себя.

1. **Интерфейс (страница настроек сервисов Telegram-бота)**

При нажатии на кнопку «Настройки» пользователь выбирает между настройками оповещений по текущему сервису или детальными настройками.

1. **Интерфейс (страница настроек частоты оповещений)**

На данном этапе пользователю предоставляется выбор, раз в какой интервал его должен оповещать бот, выбранное по умолчанию значение – 12 часов, рядом с текущим значением частоты будет зеленая галочка.

1. **Интерфейс (страница настроек рассылки новостей)**

При нажатии на кнопку «Общее» пользователь попадает на главную страницу настроек, на текущем слайде изображен интерфейс детальных настроек новостного сервиса. Здесь пользователь может регулировать, включены или нет фоновые оповещения, задавать отслеживаемые категории новостей, а также задавать список источников, откуда пользователь не хочет видеть информацию.

1. **Интерфейс (страница настроек сервиса Twitter)**

Детальные настройки раздела Twitter схожи с настройками новостей, однако тут задают никнеймы отслеживаемых пользователей в социальной сети Twitter и отслеживаемые хэштеги.

1. **Интерфейс (страница настроек рассылки погоды)**

Детальные настройки раздела погоды позволяет дополнительно настроить список городов, по которым пользователь будет смотреть текущий прогноз. При нажатии на кнопку «Локация» на мобильном устройстве будет предложено пользователю поделиться текущей геолокацией. С помощью сервиса Яндекс.Геокодер приложение определит город и добавит его в список отслеживаемых.

1. **Интерфейс (страницы модификаций пользовательских фильтров)**

На слайде представлен интерфейс добавления какого-либо пользовательского фильтра. Сначала пользователю показывается текущий список условий, и он имеет возможность добавить или убрать каждое из условий.

1. **Интерфейс (страницы просмотра информации)**

Сейчас вы можете видеть коллаж из 3-ех возможных интерфейсов просмотра информации. Здесь изображены как просмотр новостей, так и просмотр твитов соц. сети Twitter и прогноз погоды. Пользователь может свободно переходить к следующим и предыдущим элементам, также сбрасывать историю просмотра с помощью кнопки «В начало», которая покажет самую актуальную информацию. Также есть кнопка «Настройки». Если рядом с ней зеленая галочка – значит пользовательские настройки учтены при просмотре контента, если красный крестик – то идет просмотр всей информации, хранящейся в БД. Для раздела погоды есть уникальная кнопка «Локация», по нажатию на нее у пользователя запросят его геолокацию и по ней покажется информация по погоде.

1. **Интерфейс (главная страница сайта)**

На данном слайде можем видеть главную страницу сайта. Здесь отображаются в форме таблицы новости, отсортированные по количеству просмотров в разрезе времени, выбранным пользователем (по умолчанию – 10 минут). При клике на ссылку новости, кол-во просмотров возрастет, при этом обновлении информации в таблице происходит в фоновом режиме, т.е. нет необходимости перезагружать страницу каждый раз для получения новых данных.

1. **Средства реализации**

Для реализации проекта использовались средства, представленные на слайде.

1. **Требования к аппаратному и программному обеспечению сервера**

Требования к аппаратному и программному обеспечению сервера указаны на слайде.

1. **Требования к аппаратному и программному обеспечению клиента**

Требования к аппаратному и программному обеспечению клиента указаны на слайде.

1. **Логическая модель данных**

При использовании данной логической модели данных уровня сущностей была построена физическая модель данных, которая находится в приложении дипломной работы и вашем раздаточном материале (если вощинская одобрит эту затею).

1. **Структура приложения**

На данном слайде показано, как взаимодействуют друг с другом различные части приложения. Есть одна БД, один сервер, который обсуживает 2 вида клиентов: Telegram-бот и веб-страницу.

1. **Структура приложения (MVC-шаблон)**

Приложение построено на основе архитектурного паттерна MVC. Это означает, что такие аспекты как логика ввода, логика сервера и логика UI разделены между собой, но сохраняется свободная связь между ними.

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем), просто предоставляя доступ к данным и управлению ими.

Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «[наблюдателей](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))».

Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

В самом начале входящий запрос попадает на контроллер на клиенте (для нас он не виден). Далее он перенаправляет запрос в контроллер на сервере. После обработки запроса на сервере, посылается ответ к контроллеру на клиенте, и после этого результат используется для отображения представления.

1. **Схема навигации по Telegram-боту**

На текущем слайде представлена схема навигации по Telegram-боту на уровне разделов.

1. **Схема навигации по каждому из сервисов Telegram-бота**

Поскольку каждый из разделов имеет одинаковую схему навигации, то на слайде изображен общая схема навигации, применимая к каждому из реализованных сервисов.

1. **Результаты работы**

В результате была создана и протестирована веб-приложение, представляющее собой новостной агрегатор, получающий информацию как через открытое API, так и посредством парсинга RSS-каналов, предоставляющее следующие возможности:

* возможность просмотра самых свежих новостей;
* добавление и удаление категорий новостей, по которым пользователь будет получать обновления;
* возможность блокировать и снимать блокировку с новостных ресурсов;
* возможность просмотра самых новых твитов социальной сети Twitter;
* добавление и удаление отслеживаемых пользователей Twitter и хэштегов;
* возможность просмотра текущей погоды в указанных пользователем населенных пунктах;
* добавление городов в список отслеживаемых населенных пунктов;
* добавление и удаление городов для получения прогноза погоды;
* возможность просмотра новостей, твитов и погоды применяя пользовательские настройки, так и без них;
* включение и отключение оповещений, настройка их частоты отдельно для каждого из сервиса.

1. **Результаты работы**

Большое спасибо за внимание, теперь я готов ответить на ваши вопросы.