**Текст для презентации**

1. **Слайд**:

**Обоснование разработки автоматизированной системы управления данными и задачами и выбор основы для ее построения**

1. **Слайд. Контекст, проблемы**:

В процессе работы над проектами мы с коллегами сталкивались с различными проблемами, был проведён их анализ и выявлены основные:

1. Данные для анализа и разработки подготавливаются в ручном режиме и хранятся в бумажном виде (до 20%), что существенно замедляет процесс.
2. Данные передаются в формате файлов (различного расширения) разнородного вида, что приводит к необходимости написания специальных парсеров для получения данных из файла и приведения их к определённому формату. И так под каждый файл индивидуально. (В этом есть определённый смысл, но только если один раз обработать много файлов скопом и сложить данные из них в базу данных). А если формат файлов изменится? Проделанная работа не будет иметь никакого смысла.
3. **Слайд. Контекст, проблемы**

Так же есть инженеры-расчётчики в отделе, и они часто сталкивался с такими проблемами с данными как:

1. Дублирования данных,
2. Устаревшей информацией
3. Сложностями в координации с коллегами по вопросам данных.
4. **Слайд. Цели**:

Исходя из общих проблем, появились общие цели. Сформировалась команда, состоящая из Паши, Миши и меня.

1. **Слайд. Решение. Альтернативы и их недостатки:**

**1.** **Использование локальных файлов и папок, регламентов и правил обращения с ними:**

**Описание:** Установление строгих регламентов и правил для работы с данными, включая стандарты заполнения, хранения и передачи информации.

**Недостатки:**

* Человеческий фактор: Регламенты могут нарушаться или игнорироваться, особенно при высокой загруженности сотрудников.
* Отсутствие автоматизации: Процессы остаются ручными, что увеличивает вероятность ошибок и снижает скорость работы.
* Сложность контроля: Трудно отслеживать соблюдение правил в реальном времени, что может привести к нарушениям и потерям данных.
* Риск потери данных: Высока вероятность потери данных из-за ошибок пользователей или технических сбоев.
* Отсутствие версионирования: При изменении файлов сложно отслеживать историю изменений и возвращаться к предыдущим версиям.
* Ограниченная доступность: Доступ к данным ограничен, что затрудняет совместную работу и координацию.

**2. Облачные решения для хранения данных:**

**Описание:** Использование облачных сервисов, таких как Google Drive или Dropbox, для хранения и совместного использования данных.

**Недостатки:**

* Безопасность: Передача конфиденциальных данных через облачные сервисы может не соответствовать корпоративным стандартам безопасности.
* Ограниченные возможности интеграции: Облачные решения могут не поддерживать автоматизацию и интеграцию с другими системами, такими как CI/CD.
* Зависимость от интернет-соединения: Для работы требуется постоянное подключение к интернету, что может быть проблематично в случае сбоев.

**3. Специализированные программные решения:**

**Описание:** Использование специализированного программного обеспечения для управления проектами и данными, такого как Microsoft Project или Asana.

**Недостатки:**

* Стоимость: Лицензирование и внедрение специализированного ПО может быть дорогостоящим.
* Кривая обучения: Потребуется обучение персонала для работы с новым инструментом, что может занять время и ресурсы.
* Ограниченная функциональность: Не все решения поддерживают все необходимые функции, такие как контроль версий или автоматизация расчётов.

**4. Самописные системы:**

**Описание:** Разработка собственной системы для управления данными и проектами полностью с нуля.

**Недостатки:**

* Время и ресурсы на разработку: Создание и поддержка собственной системы требует значительных временных и финансовых вложений.
* Поддержка и обновления: Необходимость постоянного обновления и поддержки системы, что требует наличия квалифицированного персонала.

1. **Слайд. Контекст, идея:**

Возникла развилка:

Во время обсуждений появилось предположение, что можно экстраполировать опыт и лучшие практики организации работы с данными IT компаний на отдел расчётов.

**Использование СКВ**

Система контроля версий — это инструмент, который помогает отслеживать изменения в файлах и проектах. Вот основные идеи:

1. **Сохранение изменений**: Каждый раз, когда вы меняете файл, система сохраняет новую версию, не удаляя старые. Это позволяет видеть историю изменений.
2. **Работа в команде**: Несколько человек могут работать над одним проектом одновременно. Система следит за тем, чтобы изменения разных людей не пересекались и не конфликтовали.
3. **Восстановление предыдущих версий**: Если что-то пошло не так, можно легко вернуться к любой предыдущей версии файла или проекта.
4. **Удобное управление**: Можно создавать ветки для экспериментов или новых функций, а потом объединять их с основным проектом, когда они готовы.

**CI/CD** (Непрерывная интеграция и непрерывное развёртывание):

Непрерывная интеграция (Continuous Integration):

* Регулярное объединение изменений в работу других команд.
* Автоматическое тестирование и проверка кода для быстрого обнаружения и исправления ошибок.

Непрерывное развёртывание (Continuous Deployment):

* Автоматическое внедрение проверенных изменений в рабочую среду.
* Обеспечивает постоянное обновление системы без задержек.

Преимущество для нас: Ускоряет цикл разработки и внедрения, повышая оперативность и качество нашей работы.

Идея крутая, а кто же будет её реализовывать?

1. **Слайд. Почему именно своими силами**:
2. Изнутри мы лучше понимаем особенности работы, задачи, потребности и внутренние процессы. Кто-то со стороны не будет учитывать все нюансы, что приведёт к необходимости дополнительных правок и переработок
3. Внутренняя команда может быстро адаптировать систему под изменяющиеся задачи и новые требования, не завися от графика и приоритетов сторонних разработчиков. Возможность вносить изменения в процессе разработки без задержек, связанных с внешними коммуникациями.
4. Собственная команда может поддерживать и развивать систему в долгосрочной перспективе без необходимости трат на внешнее сопровождение.
5. В процессе разработки используются данные, которые не должны покидать пределы предприятия.
6. Готовые решения могут потребовать значительных затрат на лицензирование и кастомизацию под нужды предприятия.
7. **Слайд. Решение:**

Решение состоит из трёх основных частей.

1. **Слайд. Решение. Базы данных:**

**Логика 1 шаг**

Отказаться от мышления «файлами» и их хранения, потому что, с файлами существует ряд проблем перечисленных выше и обычно, из целого файла важны только определённые его цифири, которые и нужно собирать. Таким образом, перейти на мышление «базами данных», где каждая таблица является или исходными данными для следующих расчётов, или результатами предыдущих, с понятными источниками данных и связанных в общую цепочку причинно-следственных связей.

1. **Слайд. Решение. Система контроля версий:**

**Логика 2 шаг**

Но ведь и в базу данных можно вписать что-то не то или как-то ещё её испортить. А вот и нет, если пользоваться системами контроля версий (СКВ), которая позволяет:

1. Cоздавать и хранить версии проекта
2. Вносить изменения в проект только по согласованию
3. иметь доступ в реальном времени к любой стадии проекта
4. Возможность создавать новые ветки с измененной геометрией для «тестовых» расчетов с последующим объединением с основной геометрией
5. **Слайд. Решение. Веб сервисы:**

**Логика 3 шаг**

Любой повторяющийся расчёт можно представить в форме сервиса с собственным API, которому на вход будут поступать данные из базы и он будет складывать результаты в неё же.

Сервисы обладают рядом преимуществ перед другими форматами:

1. Отсутствие локальных версий ИД и результатов расчета, все хранится в одном месте
2. Возможность внесения обновлений без необходимости беспокоить пользователей
3. Настраивать все так как это нужно нам
4. **Слайд. Решение. Система:**

**Логика 4 шаг**

Если у нас уже есть наполненные базы данных с понятными источниками данных, и система контроля версий с широким фунционалом, почему бы не пойти дальше. СКВ имеет свой уже написанный API, который позволяет производить множество различных автоматических взаимодействий с системой (даже готовая библиотека на Python). А Таким же образом эти сервисы могут взаимодействовать между собой. На выходе получается система из базы данных и сервисов-калькуляторов, которые взаимодействуют через систему контроля версий.

**Новая система на основе GitLab**:

1. **Единый доступ к данным**: Все данные хранятся в GitLab, к которому инженер-расчётчик имеет доступ через удобный веб-интерфейс.
2. **Удобный интерфейс GitLab**: инженер-расчётчик использует интуитивно понятный интерфейс GitLab для работы с файлами, запуска расчётов и отслеживания изменений.
3. **Автоматизация расчётов через CI/CD**:
   * При изменении данных GitLab автоматически запускает расчёты через настроенные пайплайны CI/CD.
   * Результаты расчётов сохраняются в GitLab и доступны всем участникам команды.
   * Прописать что будут роли и любые данные могут быть подвержены цензуре в зависимости от роли?
4. **Контроль версий с Git LFS**: Большие файлы и бинарные данные версионируются с помощью Git LFS, позволяя инженеру отслеживать изменения и откатываться к предыдущим версиям при необходимости.
5. **Совместная работа и прозрачность**:
   * Коллеги могут одновременно работать над проектом, видеть изменения друг друга в реальном времени.
   * Обсуждения и комментарии к изменениям ведутся прямо в GitLab, упрощая коммуникацию.
6. **Автоматическое формирование отчётов**:
   * Система автоматически генерирует отчёты по итогам расчётов и публикует их в GitLab Pages или отправляет заинтересованным лицам.
   * Шаблоны отчётов хранятся в репозиториях GitLab и версионируются.
7. **Слайд. Система. Профит:**

**Преимущества**:

* **Сокращение времени на организационные задачи**: Инженер-расчётчик больше не тратит время на поиск актуальных данных и настройку окружения.
* **Повышение точности и уменьшение ошибок**: Автоматизация процессов и единая система контроля версий снижают риск человеческих ошибок.
* **Улучшение коммуникации и совместной работы**: GitLab предоставляет инструменты для эффективного взаимодействия между сотрудниками.
* **Прозрачность процессов**: Руководство может отслеживать прогресс работ, изменения и результаты, что помогает в управлении проектами.

1. **Слайд. Заключение**:

Внедрение GitLab как центральной системы для хранения данных и управления процессами изменило подход отдела расчётов к работе. Акцент сместился с локальных расчётов и файлов Excel на централизованные, автоматизированные расчёты и централизованные, автоматически обновляемые базы данных. Это позволяет оптимизировать рабочие процессы, повысить качество работы и обеспечить конкурентоспособность предприятия.

1. **Слайд. Спасибо за внимание**: