Добрый день, уважаемые коллеги!

Данным письмом я продолжаю серию писем, связанным с организацией создания информационных систем и разработкой программного обеспечения в отделе 08402 в условиях импортозамещения, а также с организацией процесса разработки программного обеспечения с использованием отечественных средств разработки.

1. **Используемые средства разработки**

В настоящее время мы используем средство разработки Oracle Application Express (<https://ru.wikipedia.org/wiki/Oracle_Application_Express>). Данное средство разработки позволяет создавать трехуровневые Web-приложения баз данных и обладает рядом несомненных достоинств:

* использование одного языка программирования Oracle PL/SQL для разработки как клиентской, так и серверной части программного обеспечения;
* наличие большого количества мастеров и редакторов, позволяющих создавать в визуальном режиме разнообразные формы, отчеты и диаграммы;
* встроенная система авторизации, аутенфикации и управления правами и доступом;
* наличие непосредственно в составе средства разработки системы контроля версий и репозитария исходного программного кода.

Наличие непосредственно в составе средства разработки системы контроля версий, репозитария исходного программного кода и системы автоматической сборки проекта является решающим преимуществом данного средства разработки, поскольку позволяет обеспечить командную работу с участием большого числа разработчиков программного обеспечения без привлечения дополнительных средств обеспечения командной разработки.

При этом:

* все изменения в программном коде, выполненные одним разработчиком, становятся незамедлительно доступными прочим разработчикам, участвующим в проекте;
* все изменения в программном коде, выполненные одним разработчиком, незамедлительно компилируются и становятся доступными для использования при следующем запуске приложения.

Данные преимущества позволяют обеспечить быструю разработку программного обеспечения, без привлечения дополнительных средств, связанных с обеспечением и синхронизацией командной разработки и автоматической сборки программного обеспечения из программных компонентов, разработанных разными разработчиками, участвующими в проекте.

Таким образом, данное средство разработки неявно обеспечивает процесс непрерывной интеграции (Continuous Integration - <https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F>) и непрерывной доставки при разработке программного обеспечения (Continuous Delivery - <https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0>). Процесс непрерывной интеграции является обязательной частью процесса непрерывной доставки либо работает совместно с ним (см. <https://docs.aws.amazon.com/codepipeline/latest/userguide/concepts-continuous-delivery-integration.html>).

При переходе на отечественные средства разработки необходимо будет организовать процесс непрерывной интеграции и доставки с использованием ряда вспомогательных средств, так как продукты, аналогичные Oracle Application Express, в настоящее время отсутствуют на рынке отечественного программного обеспечения.

Далее рассматриваются отечественная платформа для разработки программного обеспечения, язык программирования, с использованием которого мы планируем разрабатывать программное обеспечение, графическая среда разработки программного обеспечения, фреймворки, библиотеки, система контроля версий и средства, позволяющие организовать процесс непрерывной интеграции и доставки при разработке программного обеспечения и прочие вопросы, связанные с поддержкой эффективной разработки программного обеспечения.

* 1. **Платформа для разработки программного обеспечения**

Для разработки программного обеспечения планируется использование платформы Jmix (ранее – Cuba.Platform) (<https://reestr.digital.gov.ru/reestr/462834/?sphrase_id=1669211>). С основными характеристиками платформы можно ознакомиться здесь: <https://www.jmix.ru/framework/>. Так же рекомендуется к просмотру данный YouTube канал, демонстрирующий разработку программного обеспечения с использованием данной платформы: <https://www.youtube.com/c/JmixFramework>.

В настоящее время мы обладаем лицензиями на 4-ех разработчиков, таким образом, нам необходимо определить перечень сотрудников отдела, которые непосредственно будут работать с платформой. Я предлагаю собрать следующую команду: Сермин Н.А., Дмитрущенков В.А., Красавцев К.Ю., Макаров А.Е. В случае, если будут достигнуты определенные успехи, на будущий год я запланирую расширение количества лицензий до 6-ти.

* 1. **Язык программирования**

Для разработки программного обеспечения будет использоваться язык программирования Java (<https://ru.wikipedia.org/wiki/Java>).

* 1. **Графическая среда разработки**

В качестве среды разработки используется отечественная среда разработки Jmix Studio (<https://reestr.digital.gov.ru/reestr/418536/?sphrase_id=1669211>). Данное программное обеспечение представляет собой плагин (<https://www.jmix.ru/tools/>) к распространяемой свободно среде разработки IntelliJ IDEA Community Edition (<https://www.jetbrains.com/idea/>).

* 1. **Фреймворки, библиотеки**

Платформа Jmix представляет собой фреймворк и набор библиотек, базирующийся на таких распространенных фрейворках и библиотеках как EclipseLink (<https://ru.wikipedia.org/wiki/EclipseLink>), Spring Framework, (<https://ru.wikipedia.org/wiki/Spring_Framework>), Spring Boot (<https://topjava.ru/blog/introducing-spring-boot>) и Vaadin Flow Framework (<https://vaadin.com/flow>). Работа с платформой Jmix не требует знаний данных фрейворков и библиотек, однако знакомство с идеологией этих фрейворков крайне желательно. Данные статьи (<https://spring.io/guides/gs/crud-with-vaadin/> и <https://vaadin.com/docs/v14/flow/tutorial/overview>) демонстрируют процесс использования этих фреймворков для создания законченного Web-приложения.

Так же мы можем использовать любые фреймворки и библиотеки, которые обеспечивают функциональность, которую нам необходимо обеспечить в разрабатываемом нами программном обеспечений. Большинство фреймворков и библиотек Java доступны бесплатно из репозитариев Maven на сайтах разработчиков либо из централизованного репозитория Maven на сайте MVN Repository (<https://mvnrepository.com/>), при этом эти репозитарии позволяют использовать зависимости (dependencies), которые могут быть прописаны в сборочном файле проекта. При этом при сборке проекта система автоматической сборки проекта автоматически выполнит в удаленных репозиториях поиск и загрузку необходимых библиотек на компьютер разработчика с последующей интеграцией этих библиотек в состав разрабатываемого программного обеспечения (про систему автоматической сборки см. пункт е) на этапе автоматической сборки проекта.

* 1. **Система автоматической сборки**

Подробнее про автоматизацию сборки см. <https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B8>.

Система автоматической сборки использует так называемый сборочный файл, в котором разработчик определяет, как и с использованием каких библиотек необходимо собрать проект.

Среда разработки Jmix Studio использует систему автоматической сборки Gradle (<https://ru.wikipedia.org/wiki/Gradle>). При этом загрузка, настройка и установка этой системы автоматической сборки не требуется. Максимум что может потребоваться – это прописать в сборочном файле какие-либо зависимости, не входящие в платформу Jmix по умолчанию с тем, чтобы система сборки могла осуществить загрузку этих библиотек. Подробнее про работу с Gradle: <https://www.youtube.com/playlist?list=PLnh8EajVFTl7JYxhZs7CfxyTsUMvcBXig>

* 1. **Собственный репозиторий Maven**

Использование централизованного репозитария Maven или репозитариев разработчиков на сайтах разработчиков программного обеспечения сопряжено с рядом проблем с безопасностью таких как:

1. Необходимостью наличия постоянного подключения компьютера разработчика программного обеспечения к сети Интернет.
2. Возможностью внесения деструктивных команд в код, размещенный в одном из этих центральных репозитариев. В текущих условиях такой код, будучи интегрированным в разрабатываемое нами программное обеспечение, потенциально способен определить, что он функционирует на сервере, размещенном в России, и выполнить определенные действия деструктивного характера.

Таким образом, с целью обеспечения безопасности разработки, необходимо создать собственный, используемый только нами репозиторий Maven, который будет содержать только уже проверенный нами программный код и не будет подвержен внешнему воздействию с целью внесения в него деструктивных правок.

При этом управление данным репозитарием и включение в него новых библиотек Java будет являться нашей задачей. (см. так же в этих ваших Интернетах термины «железный занавес», «фундаментальное огораживание», «анальное огораживание»).

Подробнее про организацию разработки в изолированной сети с использование платформы Jmix смотри: <https://forum.jmix.ru/t/ustanovka-jmix-plagina-v-zakrytom-konture-bez-interneta/216>, <https://habr.com/ru/company/haulmont/blog/512966/>.

* 1. **Система контроля версий**

Поскольку мы планируем, что над одним проектом будет работать несколько разработчиков, для работы над проектом мы будем использовать систему контроля версий (<https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8>).

В качестве системы контроля версий мы будет использовать Git (<https://ru.wikipedia.org/wiki/Git>).

Подробнее про работу с Git: <https://www.youtube.com/watch?v=TMeZGvtQnT8>, дополнительная информация по сложным вопросам работы с Git: <https://youtu.be/kmGsHjQ2wsY>, <https://youtu.be/BF2OHMM86Ik>, <https://youtu.be/NXaEImbo-n8>.

Отмечу, что среда разработки IntelliJ IDEA Community Edition имеет в своем составе инструменты, существенно облегчающие работу с Git: <https://youtu.be/J29YZoLAZqA>.

Система контроля версий будет представлять собой репозиторий в который каждый разработчик вносит свои изменения. Данный репозиторий будет служить базой для организации процесса непрерывной интеграции и доставки, поскольку будет содержать программный код, в разработке которого приняли все участники проекта, работая каждый над своим компонентом разрабатываемого программного обеспечения. На основании содержимого репозитория будет выполняться автоматическая или ручная сборка разрабатываемого программного обеспечения.

* 1. **Средства организации процесса непрерывной интеграции и доставки**

Непрерывная интеграция (Continuous integration) – практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий проекта в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.

Непрерывная доставка (Continuous delivery) – это подход к автоматизации аспекта доставки программного обеспечения, который фокусируется на:

1. Объединении процессов сборки, тестирования и выпуска релизов программного обеспечения.
2. Выполнения данных процессов быстро и часто.

Для организации процесса непрерывной интеграции и доставки необходимо выполнение ряда базовых требований к проекту. В частности, исходный код и всё, что необходимо для сборки и тестирования проекта, должно храниться в репозитарии системы контроля версий, а операции копирования из репозитория, сборки и тестирования всего проекта должны быть автоматизированы с использованием системы автоматической сборки и легко вызываться из внешних программ.

Для организации процесса непрерывной интеграции и доставки на выделенном сервере запускается программное обеспечение, в задачи которого входят:

* получение исходного кода из репозитория;
* сборка проекта;
* выполнение тестов;
* развёртывание готового проекта;
* отправка отчетов.

Отметим, что первые три задачи представляют собой процесс непрерывной интеграции.

Сборка проекта может осуществляться по внешнему запросу, по расписанию, по факту обновления репозитория и по другим критериям.

Сборки по расписанию, как правило, проводятся в нерабочее время и планируются таким образом, чтобы к началу очередного рабочего дня были готовы результаты тестирования. При этом обычно вводится система нумерации сборок —каждая сборка нумеруется числом, которое увеличивается с каждой новой сборкой. Исходные тексты и другие исходные данные при получении их из репозитория системы контроля версий помечаются номером сборки. Благодаря этому, точно такая же сборка может быть точно воспроизведена в будущем — достаточно взять исходные данные по нужной метке и запустить процесс сборки снова. Это даёт возможность повторно выпускать даже очень старые версии программы с небольшими исправлениями.

Непрерывная интеграция и доставка является частью методологии DevOps (<https://ru.wikipedia.org/wiki/DevOps>).

Для обеспечения процесса непрерывной интеграции и доставки мы планируем использовать программное обеспечение GitLab (<https://ru.wikipedia.org/wiki/GitLab>, <https://gitlab.com/>). В дальнейшем мы будем использовать это же программное обеспечение для реализации методологии DevOps, поскольку оно поддерживает основные функции, необходимые для реализации методологии DevOps (<https://www.youtube.com/watch?v=1P3eDs9Zzyk>).

Для обеспечения процесса непрерывной интеграции и доставки необходимо создать с использованием программного обеспечения GitLab так называемый канал Continuous integration/Continuous delivery (CI/CD).

Рассмотрим этот процесс поподробнее:

1. Программисты разрабатывают исходный код программного обеспечения на своих компьютерах (в нашем случае используется отечественная среда разработки Jmix Studio).
2. Программисты разделяют исходный код программного обеспечения между собой, размещая его в системе контроля версий (в нашем случае будет использована система контроля версий Git из состава программного обеспечения GitLab). Система контроля версий (SCM) является необходимой составляющей, началом канала CI/CD.
3. Следующей частью канала является составляющая Build, обеспечивающая компиляцию исходного кода и создание пакетов и библиотек. В случае языка Java это могут быть исполняемые файлы (jar) или файлы, предназначенные для их размещения на сервере приложений (war, ear). При этом данная стадия может состоять из нескольких шагов: собственно, компиляция исходного кода, создание скриптов SQL, запуск тестов для проведения модульного тестирования, проведение статического анализа кода и пр.
4. Следующей частью канала является составляющая Release, как раз и обеспечивающая размещение (deploy) приложения в среде, предназначенной для его эксплуатации (production). При этом данная стадия может состоять из нескольких шагов: создание необходимых объектов в базе данных, размещение в тестовой среде, проведение различных тестов (User Acceptance Tests, Quality Assurance Tests и пр.), размещение в среде, предназначенной для эксплуатации.

При этом можно отметить, что составляющая Build является каналом CI, с составляющая Release – каналом CD. Запуск канала на выполнение может выполняться как вручную, так и по факту наступления определенного момента времени или события.

Так же см.: <https://youtu.be/XusC2o-Y_fU>.

Конкретный пример настройки канала CI/CD для программного обеспечения, разработанного на языке Java, приведен здесь: <https://youtu.be/ypbGz7kE-Bo>.

В директории «Обмен/GitLab» у Дмитрущенкова В.А. собраны книги по программному обеспечению GitLab. Я бы советовал для начала ознакомиться с книгой «GitLab Quick Start Guide.pdf», а затем – «Mastering GitLab 12 - Implement DevOps culture and Repository Management Solutions.pdf».

Так же неплохой практически полный курс: <https://www.youtube.com/watch?v=8aV5AxJrHDg>.

Ожидается, что в третьем квартале 2022 года данные возможности будут встроены непосредственно в среду разработки Jmix Studio с выходом их нового продукта Jmix DevOps Cloud (<https://www.jmix.ru/devops-cloud/>). Данный продукт целиком и полностью будет базироваться на GitLab. Полагаю, что мы будем использовать именно этот продукт для организации процесса непрерывной доставки и, в дальнейшем, для реализации методологии DevOps.

* 1. **Средства постановки и контроля выполнения задач**

Для постановки и контроля выполнения задач может использоваться программное обеспечение GitLab (<https://www.youtube.com/watch?v=1P3eDs9Zzyk>).

* 1. **Средства общения и документирования**

В качестве средств общения и документирования так же может использоваться программное обеспечение GitLab (<https://www.youtube.com/watch?v=1P3eDs9Zzyk>).

* 1. **Средства автоматизированного тестирования**

В настоящем планируется проводить только модульное тестирование с использованием фреймворка для модульного тестирования JUnit (<https://ru.wikipedia.org/wiki/JUnit>). В дальнейшем, по мере освоения инструментария, планируется добавление процедур интеграционного, системного и прочих видов тестирования.

1. **Организация процесса разработки**

Not ready yet