Общее описание функциональных характеристик и работы с ITA::Forms v4

IT-ALLIANCE

Оглавление

Краткое описание решения ITA::FORMS

ПО «ITA::Forms» - Интеграционная платформа для внедрения web-приложений, позволяющих выполнять в едином окне бизнес-процессы различного функционального назначения. ПО «ITA::Forms» взаимодействует с компонентами ИС заказчика с полным сохранением и поддержкой бизнес процессов. Существующие системы не заменяются, а включаются в информационную среду.

Основные свойства ПО «ITA::Forms»:

- решение основано на платформе Java JDK v8, которая является де-факто промышленным стандартом при построении многозвенных информационных систем предприятия и поддерживается как ведущими мировыми производителями программного обеспечения, так и огромным количеством независимых разработчиков. Решение позволяет выстраивать взаимодействие с внешними информационными системами с представлением функций прикладной бизнес-логики системы в виде web-сервисов на основе современных общепринятых технологий (REST) и использованием сервисной шины предприятия (ESB).
- является кросс-платформенным, что устраняет привязку к определенной операционной системе.
- реализация приложения в форме Web-интерфейса позволяет минимизировать требования к аппаратному и программному обеспечению рабочих мест пользователей. Требуется лишь наличие Web-браузера и подключение к Intranet/Internet.
- ПО создаёт широкие возможности по встраиванию в порталы других систем.

Интеграционные возможности

ПО «ITA::Forms» имеет следующие интеграционные возможности:

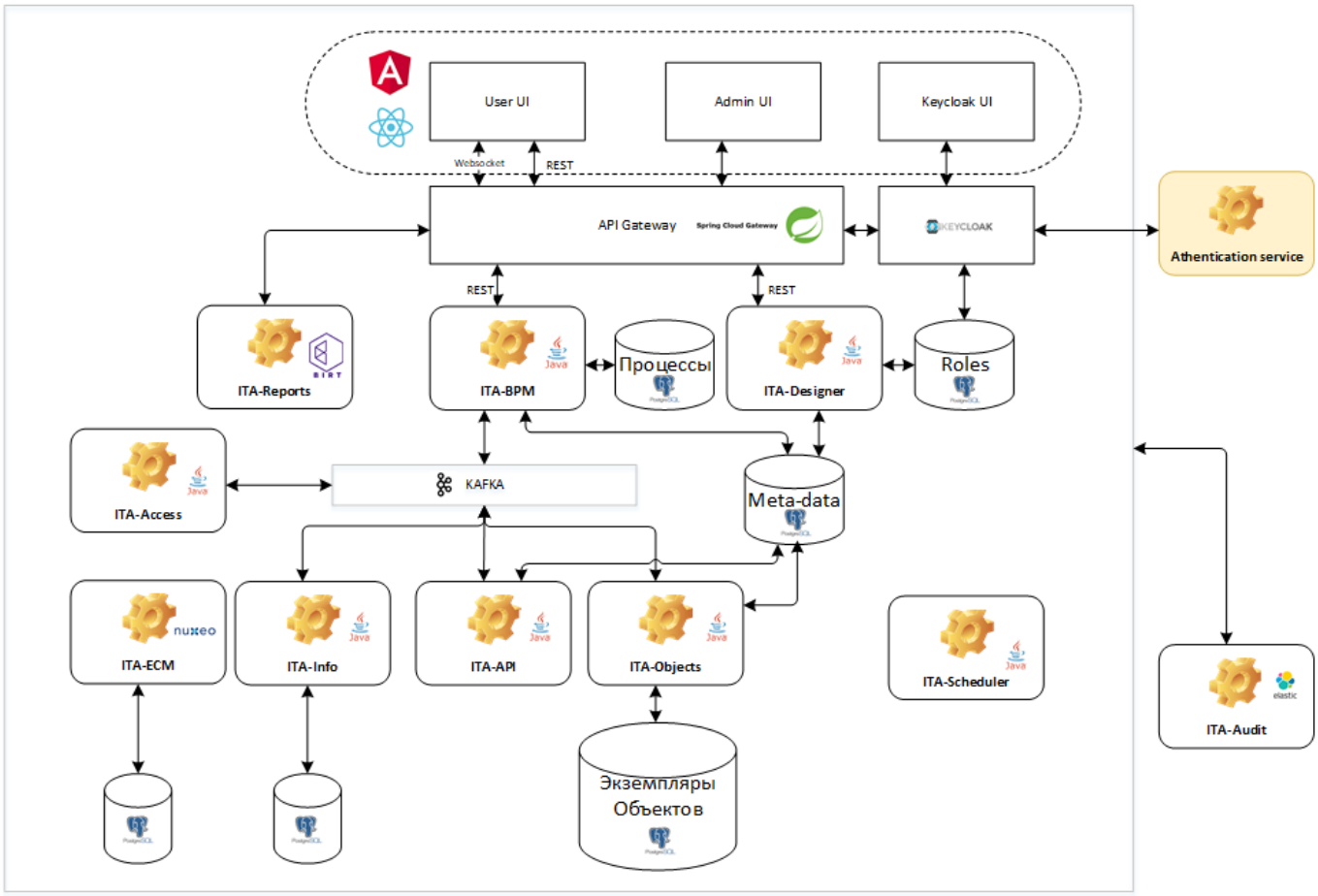
- Предоставление/вызов WEB/REST сервисов
- Использование системы очередей (RabbitMQ, Kafka и т. д.)
- Интеграция посредством библиотек (Spring)
- Файловый обмен

Ключевые преимущества решения ITA::Forms:

- Использование «ITA::Forms» - это путь к новым технологическим возможностям по модернизации и расширению функциональных задач прикладной архитектуры, стратегическому преобразованию бизнеса, к сокращению затрат на операционное и сервисное обслуживание.
- Промышленная технология модернизации и развития ИТ-инфраструктуры
- Расширение прикладной и технологической функциональности ИС
- «Единое окно» для автоматизации работы бэк-офисных сотрудников
- «Единое окно» для централизованного ввода и обработки данных физически хранимых в разных БД или АС
- Централизованное управление задачами

- Поддержка групповых и фоновых операций, интеграция с офисными приложениями
- Простая возможность масштабирования системы
- Повышение производительности за счет распараллеливания вычислений и запросов к внешним системам
- Простота развертывания и обновление новых версий
- Сохранение инвестиций для развития ИС

Архитектура платформы ITA::Forms



Платформа ITA::Forms обеспечивает создание функциональных модулей системы в формате «low-coding». Сервисы платформы позволяют оперативно осуществлять создание интерфейсов, оркестровку процессов, управление данными, распределение ролей, интеграцию с внешними системами, а также обеспечивают безопасность и аудит сервисов.

В платформу ITA::Forms входят следующие системные сервисы:

1. **Web-интерфейсы** – технология построения легкого UI на Angular и React, использующая механизмы дизайнера форм для доступа к данным и процессам.
2. **ITA::Reports** – обеспечивает получение отчётов как внутренними механизмами, так и через обращения к внешним системам. Например, сейчас через Отчётную подсистему осуществляется доступ и получение всех отчётных форм ИС или формирование отчетов с помощью внешней системы Birt.
3. **ITA::API** – обеспечивает доступ Внешним системам к данным и операциям над ними, а также приведения в соответствие классов обработки платформы и объектов внешних систем. Данная подсистема позволяет создавать web-сервисы к существующей функциональности системы. Выставлять REST сервисы для использования в других приложениях.
4. **ITA::Audit** – обеспечивает настраиваемое логирование событий, происходящих в системе, действий пользователей.
5. **ITA::BPM** – обеспечивает оркестровку и управление пользовательскими и сервисными задачами и процессами
6. **ITA::Info** – обеспечивает создание, хранение, кэширование и распространение системных констант и справочников между сервисами платформы
7. **ITA::ECM** - обеспечивает управление цифровыми документами и другими типами контента, а также их хранение, обработка и доставка в рамках организации.
8. **ITA::Access** – обеспечивает контроль вариантов доступа к бизнес-процессам, бизнес-операциям и к данным бизнес-объектов.
9. **ITA::Scheduler** – обеспечивает настройку вызова операций, процессов и сервисов по расписанию.
10. **ITA::Designer** - фреймворк для структурированной и быстрой разработки web-приложений. Фреймворк включает в себя следующие функциональные компоненты:
    - Бизнес-процессы – для описания последовательностей выполнения задач сотрудниками и сервисами организации по стандарту BPMN 2.0. Например, процессы согласования, кредитный конвейер и т.п.
    - Бизнес-объекты – для описания данных (модели данных), свойств и методов обработки бизнес-сущностей и справочников, используемых в приложении. А также настройки doc-flow (статусная модель) и ролевого доступа. Например, продукт, клиент, заявка, договор, лицевой счет, тарифный план и т.д.
    - Операции – для описания последовательности вызова сервисов, реализующих ту или иную бизнес-логику при обработке бизнес-объектов. Например, операции чтения, сохранения, обработки, …
    - Экранные формы – для описания различных элементов пользовательского интерфейса: форм ввода, элементов управления моделью данных бизнес-объектов приложения. Например, Анкета ФЛ, ЮЛ, Карточка счета, Справка

по счету, Договор РКО, Подбор вклада, Заявка на открытие вклада, Заявка на открытие счета.

- Ролевая модель (варианты доступа) – для описания вариантов доступа к бизнес-операциям и к данным бизнес-объектов.

11. **ITA::Objects** - выполняет синхронные и асинхронные действия над объектами системы и включает в себя:
    - Подсистему операций – настройка классов обработки с жизненными циклами объектов, состояниями, переходами. В частном случае является надстройкой над существующей операционной моделью документооборота ИС
    - Модели данных бизнес-объектов с настраиваемой структурой хранения и встроенными механизмами по получению и модификации данных.
    - Подсистема доступа – обеспечивает доступ к операциям над объектами системы, включает в себя в том числе и приведение в соответствие внутреннего доступа Машины операций с доступом, представляемым внешними системами. Фактически Подсистема доступа формирует Среду обработки для пользователя системы.
    - Репозиторий – хранилище для обеспечения функций Машины операций (хранение настроек классов обработки, маппинга объектов, доступа, логирования операций, пользовательских настроек, etc).

В основу ITA::Forms заложены следующие принципы:

1. Сервисная архитектура платформы. Использование независимых сервисов, выполняющих точно определённые задачи, вызываемых стандартным способом через точно определённые интерфейсы, при отсутствии у сервисов знаний о вызывающих их приложениях, а у приложений - способов, которыми сервисы выполняют свою задачу.
2. Объектно-ориентированный подход к построению сервисов.
    - классы бизнес-объектов (или просто бизнес-объекты), выстроенные в дерево наследований свойств и методов
    - методы обработки данных (или просто обработчики) с перегрузкой в дочерних классах и ролевым доступом к ним
    - модели данных с расширениями в дочерних классах и ролевым доступом к узлам модели
    - ролевая модель, накладываемая на объекты системы для дополнительного разделения методов обработки
3. Единый подход к построению любых частей системы. Это предполагает, что будь то код, описывающий обработку данных, или код, описывающий работу экранной формы – принцип его организации един и сводится к простой обработке входящих и исходящих данных. При этом всё, что нужно освоить java-разработчику для работы с системой – это встроенный в систему набор библиотек

для обработки данных. Основная сложность в освоении принципов такой разработки – это парадигма работы только с данными и ничем больше. Для принятия этого разработчику нередко приходится менять взгляд на жизнь ☺.

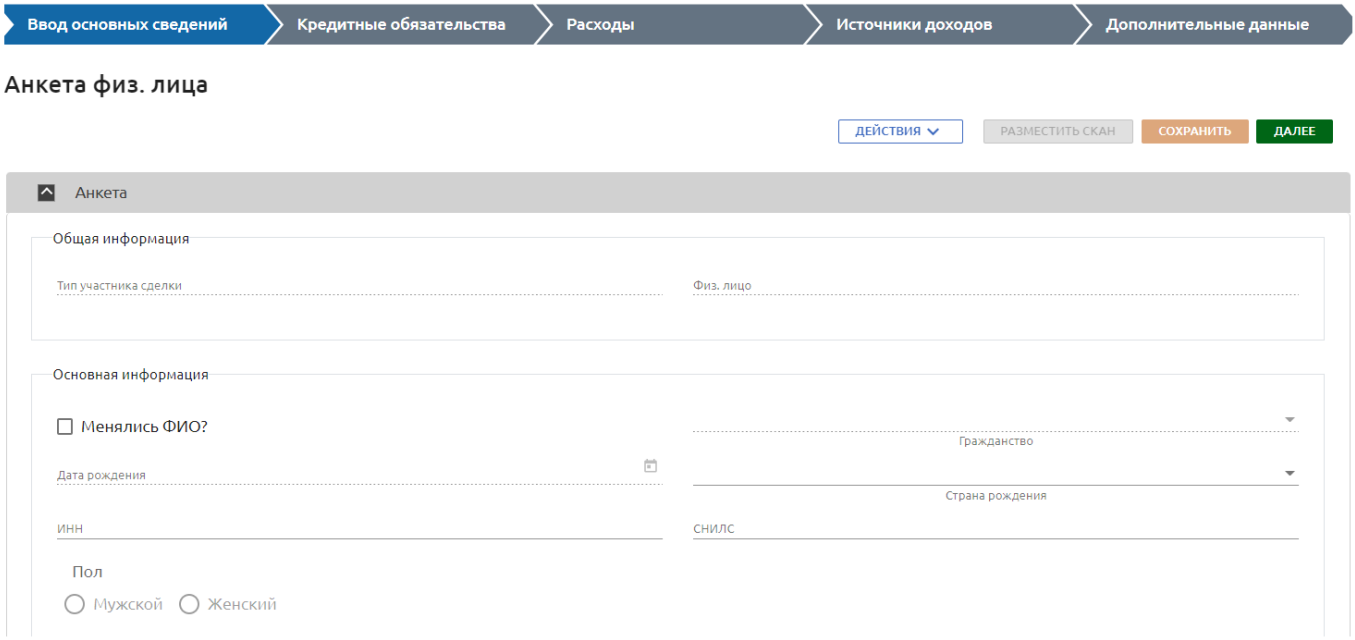
4. Компонентная организация сервисов. Предполагает полную независимость в разработке как отдельных бизнес-объектов, так и отдельных обработчиков внутри бизнес-объектов. Вся сборка системы и определение её целостности происходит во время работы приложения.
5. Скрипты и полная динамика. Все части приложения формируются в виде кода на Groovy, который оформляется в виде скриптов. Скрипты загружаются в систему в момент выполнения приложения. Это позволяет менять поведение любой части системы без изменения и даже остановки запущенного приложения.
6. Неограниченные возможности. Предполагается, что нет никаких ограничений в том, насколько сложную бизнес логику нужно реализовать, какую интеграцию с другими системами нужно осуществить. Так же нет ограничений по сложности экранных форм. Можно рисовать как простые экранные формы с несколькими контролами, так и сложные многостраничные формы с неограниченным взаимодействием их компонент.
7. Так как сервис – это java-приложение, то при разработке доступны любые библиотеки java, есть практически неограниченные возможности по организации взаимодействия с другими системами.
8. Свобода выбора как сервера приложений, так и баз данных. Требования и к одному и к другому минимальны.

Предоставляемые возможности

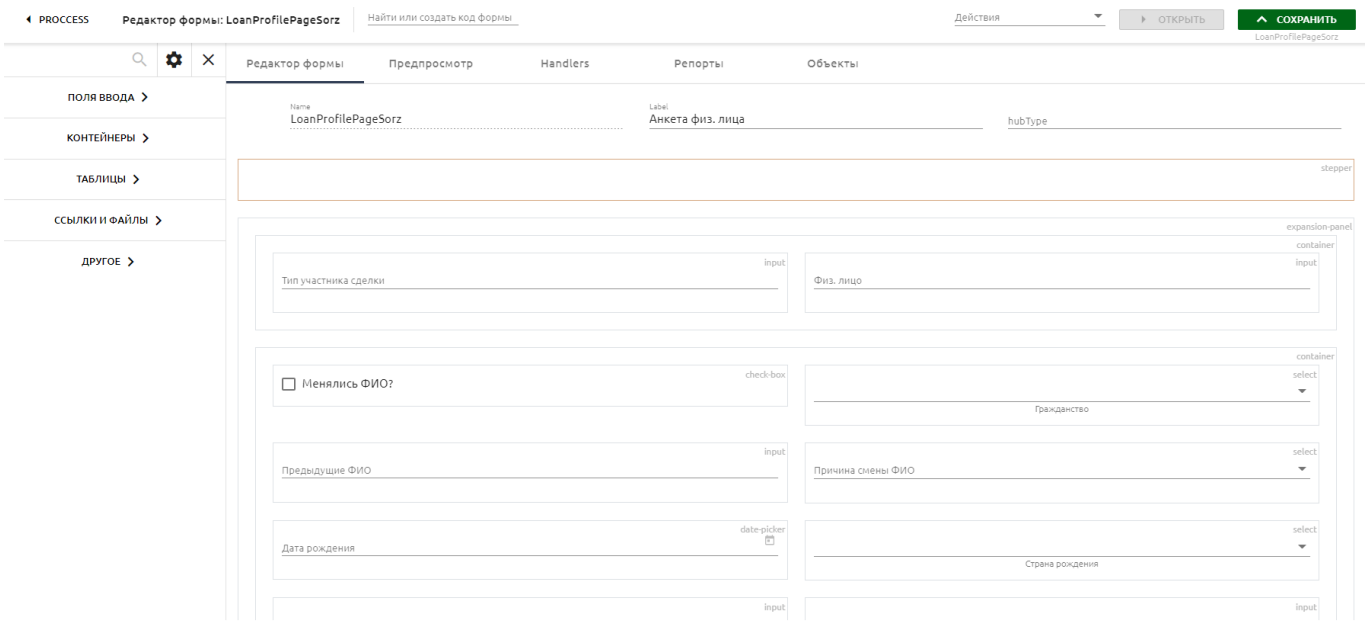
Перечислим с краткими примерами, что дает заказчику и разработчику использование ITA::Forms. Все примеры приведены из реального проекта.

1. Создавать пользовательские интерфейсы по понятным описаниям. Например, ниже представлена заполненная форма и описание её внешнего вида (далеко не самое простое). Даже новичку в web-интерфейсах через короткое время становится понятно, как это устроено:

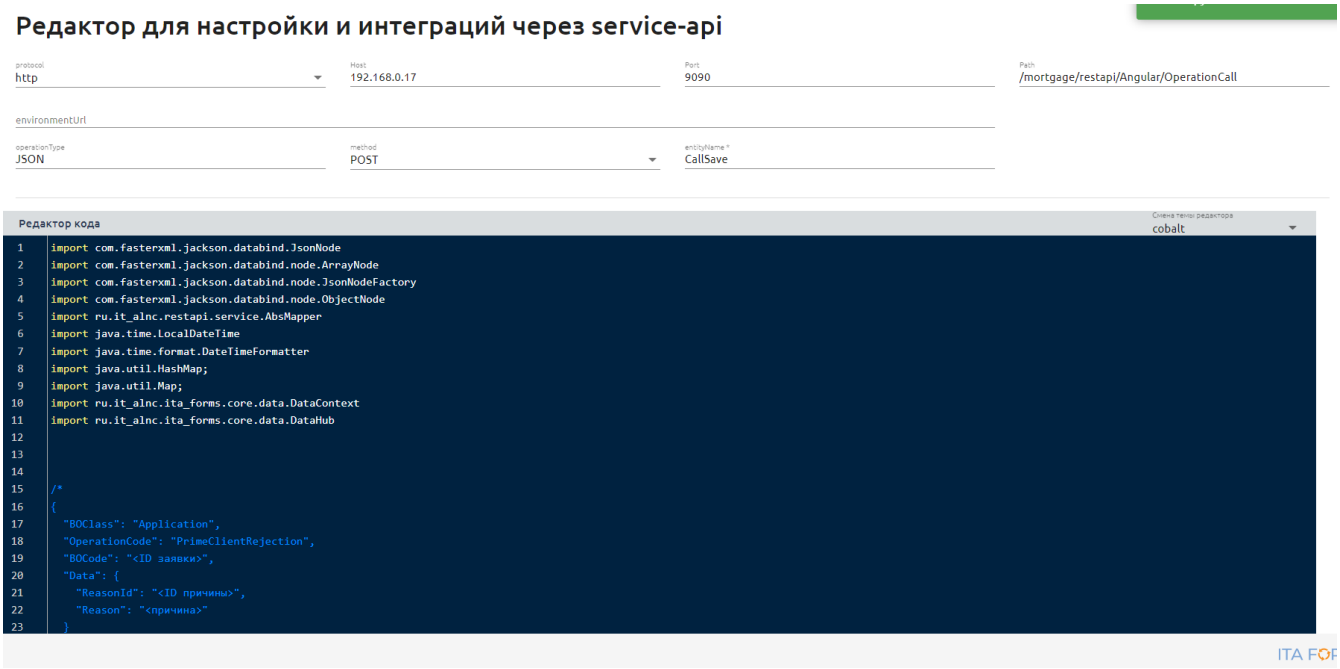
   Форма:



   Шаблон:



2. Осуществлять привязку форм и сервисов к шагам бизнес-процесса.



3. Описывать модели данных и настраивать статусный и ролевой доступ к данным. В следующем примере показано описание модели данных:

```
<!--МОДЕЛЬ Main: Модель данных заявки-->';s:=s||'
<bomodel class="Order" code="Main" label="Модель данных заявки">
  <model>
    <OrderTitle type="HIERARCHY" comment="Структура с заголовочной атрибутикой заявки">
      <SaveToABSDate type="DATE" comment="Прописать в АБС в"/>
      <ExecutionUser type="STRING" comment="Ответственный"/>
      <Result type="STRING" comment="Комментарий"/>
    </OrderTitle>
    <Attachment type="HIERARCHY" comment="Документы"/>
  </model>
</bomodel>
```

и 2-ух вариантов доступа по полям для разных статусов объекта

```
<!--ВАРИАНТ MainExecute: В обработке-->';s:=s||'
<bovariant class="Order" code="MainExecute" model="Main" label="В обработке">
  <status>
    <Execute/>
  </status>
  <model>
    <OrderTitle type="HIERARCHY" access="READ">
      <Result type="STRING" access="EDIT"/>
      <SaveToABSDate type="DATE" access="EDIT"/>
    </OrderTitle>
    <Attachment type="HIERARCHY" access="READ"/>
  </model>
</bovariant>
<!--ВАРИАНТ MainNew: Новый-->';s:=s||'
<bovariant class="Order" code="MainNew" model="Main" label="Новый">
  <status>
    <New/>
  </status>
  <model>
    <OrderTitle type="HIERARCHY" access="EDIT">
      <ExecutionUser type="STRING" access="READ"/>
    </OrderTitle>
  </model>
</bovariant>
```

4. Описывать бизнес-операции и формы, настраивать доступ к ним. Бизнес-операции создаются посредством описания их свойств, параметров и последовательности обработчиков с программированием бизнес-логики, когда это нужно.

   Например, описываем операцию “Удалить строку тарифного плана”:

```
<booperation class="OrderTariff" code="DropTariffPlanLine" label="Удалить строку тарифного плана
```

   и даём доступ к ней пользователям с ролью “User” в состоянии “New”, а с ролью “Manager” в состоянии “Opened”:

```
<role>
  <User status="New"/>
  <Manager status="Opened"/>
</role>
```

   Доступ к формам настраивается аналогичным образом.

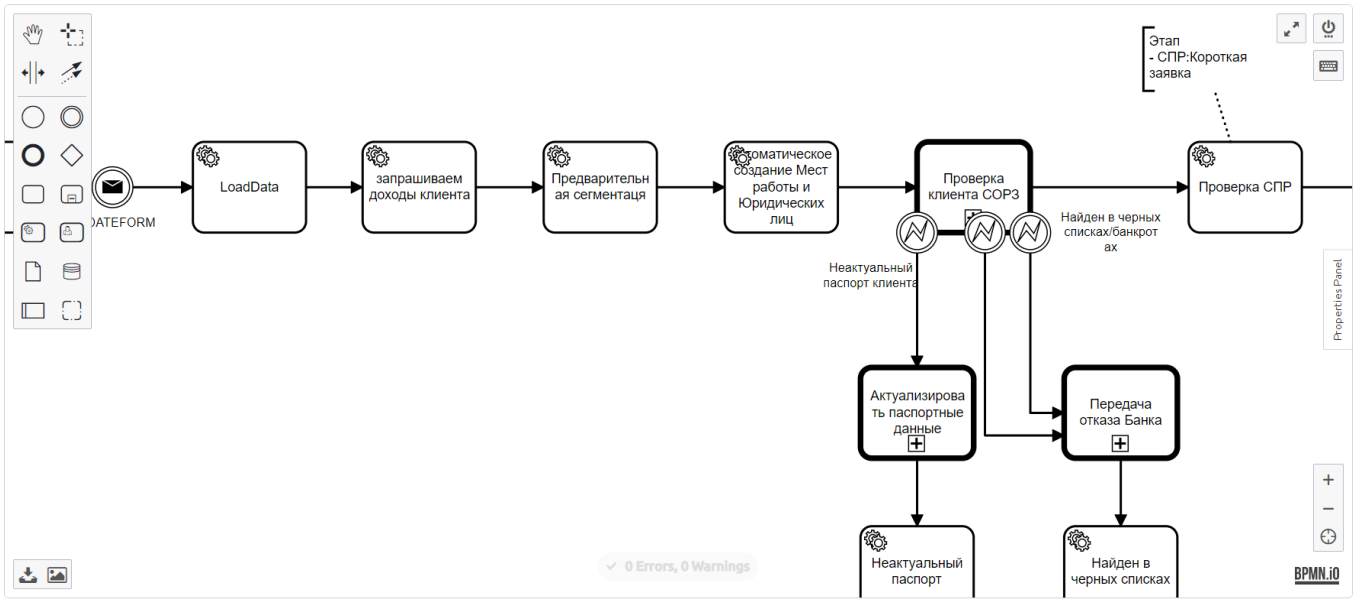
   Таким образом можно настраивать доступ к любым операциям и экранным формам и гарантированно ограничивать пользователей от недопустимых действий.

Процесс разработки

Для лучшего представления о том, как использовать фрэймворк ITA::Forms в разработке нового функционала, рассмотрим пример разработки бизнес-процесса подбора и открытия Вклада. Разберем процесс разработки по шагам.

Шаг 1 – дизайн бизнес-процесса

На этом шаге создается общая схема процесса BPM.



Для отображения процесса в системе ITA::Forms создаем следующие бизнес-объекты

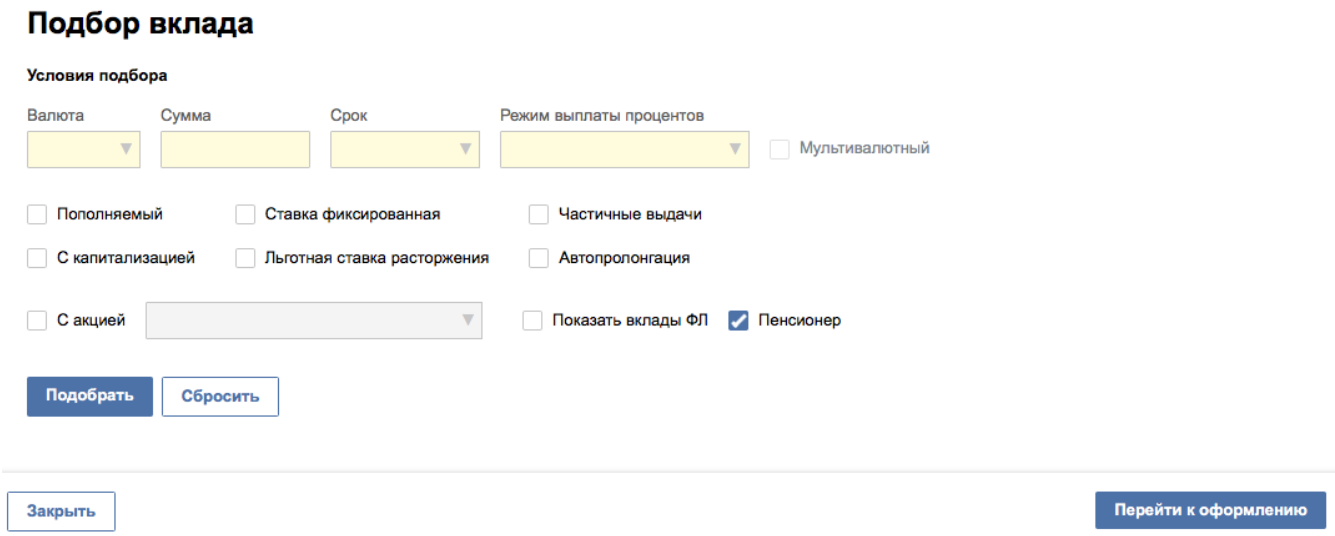
- Депозитный продукт
- Заявка на открытие вклада
- Вклад

Шаг 2 – определение перечня экранные формы для пользовательских действий процесса

На этом шаге необходимо определить и нарисовать набор необходимых форм (в бизнес-объектах). Для представленного ранее БП это:

- Подбор Вклада (Депозитный продукт)
- Условия Вклада (Заявка на открытие вклада)
- Замена Вложения (Заявка на открытие вклада)
- Открытие Вклада (Заявка на открытие вклада)
- Обработка Ошибок (Заявка на открытие вклада)
- Просмотр вклада (Вклад)

Например, для формы «Подбора вклада» определяем, что выглядить она будет так:



Шаг 3 – определение операций бизнес-логики процесса

На этом шаге, исходя из схемы BPM, необходимо определить набор операций над бизнес-объектами, вызов которых надо будет настроить в BPM. А так же набор вспомогательных операций, используемых в формах. В нашем примере это:

- Начать процесс (Заявка на открытие вклада)
- Завершить процесс (Заявка на открытие вклада)
- Прочитать условия для подбора Вклада (Депозитный продукт)
- Создать Заявку (Заявка на открытие вклада)
- Прочитать Заявку (Заявка на открытие вклада)
- Открыть Вклад (Заявка на открытие вклада)
- Открыть Вклад VIP (Заявка на открытие вклада)
- Создать Вклад (Вклад)
- Прочитать Вклад (Вклад)
- Прочитать Ошибки (Заявка на открытие вклада)
- Отбраковать Заявку (Заявка на открытие вклада)

Шаг 4 – описать модели данных

В нашем примере описываются 2 модели данных – для Заявка на открытие вклада и для Вклада. Для этих моделей настраиваются Варианты доступа, через которые ролям раздаются доступы к тем или иным данным бизнес-объекта. Например, Контролеру дается доступ только на просмотр данных Заявки, а Клиентскому менеджеру – возможность редактирования Заявки.

Шаг 5 – программирование логики операций, определенных на шаге 3.

Шаг 6 – программирование логики форм, определенных на шаге 2.

На этом шаге связывается воедино созданные формы и операции, программируется дополнительное (отличное от стандартного) поведение контролов форм.

Шаг 7 – запуск процесса и отладка

Запуск и отладка приложения производится на общем сервере. Это означает, что несколько независимых групп разработчиков осуществляют отладку различного функционала в единой среде. В процессе отладки разработчик производит модификацию кода, добавляет отладочную информацию, трассирует выполнение отдельных операций и целых бизнес-процессов. Всё это происходит в окне браузера без необходимости иного (чем через браузер) взаимодействия с сервером приложений.

Требования к разворачиванию приложений на платформе

Программные продукты «ITA::Forms» предназначены для разворачивания и эксплуатации в программно-аппаратной конфигурации.

Варианты разворачивания приложений на платформе «ITA::Forms»:

Тип конфигурации	Состав конфигурации
СПО	БД: PostgreSQL ОС с поддержкой Docker Kafka

Системные требования к серверам приложений и БД для разворачивания.

Базовое ПО (минимальные требования)		
Elasticsearch (запуск внутри системы оркестрации контейнеров)	сервер, шт	1
	cores xeon*3GHz	2
	RAM GB	6
	HDD	100
Kafka (запуск внутри системы оркестрации контейнеров)	сервер, шт	1
	cores xeon*3GHz	2
	RAM GB	4

	HDD	100
PostgreSQL	сервер, шт	1
	cores xeon*3GHz	16
	RAM GB	32
	HDD	100

Базовое ПО (рекомендованные требования)		
Elasticsearch	сервер, шт	1
	cores xeon*3GHz	4
	RAM GB	8
	HDD	100
Kafka	сервер, шт	3
	cores xeon*3GHz	4
	RAM GB	4
	HDD	100
PostgreSQL	сервер, шт	2
	cores xeon*3GHz	32
	RAM GB	64
	HDD	100

Системные требования к рабочим станциям

OS Windows 7/8/10, или UNIX-совместимые с установленным браузером Google Chrome, Mozilla Firefox, Microsoft Internet Explorer (не ниже версии 10)	Реализация приложений в форме Web-интерфейса позволяет минимизировать требования к аппаратному и программному обеспечению рабочих мест пользователей. Требуется лишь наличие Web-браузера и подключение к Internet