Автоматизация процессов обработки кредитной заявки в банке

Профиль пользователя:

Российский банк

Функциональная область:

Разработка ПО

Оказанные услуги:

Разработка системы, тестирование

Длительность:

ноябрь 2011- февраль 2012

Технологии и средства разработки

- Spring MVC
- JSP
- JavaScript
- PostgreSQL
- Hibernate
- Activity
- JUnit
- SpringTest
- Selenium

Проблема

Заказчиком выступал региональный банк с отделениями во многих городах европейской части России. Руководством Банка с целью повышения конкурентоспособности в своем сегменте рынка было принято решение автоматизировать бизнес-процесс обработки кредитной заявки. Процесс включает в себя следующие роли и этапы:

1. Кредитный Эксперт (КЭ) принимает заявление клиента на кредит, заполняет данные, прикладывает сканы необходимых документов. Заявка отсылается МФА.
2. Менеджеру по Финансовому Анализу (МФА) рассматривает заявку, или отказывает, или одобряет или отправляет на заявку доработку.
3. КЭ подписывает договор с клиентом, если она одобрена
4. Бухгалтерия перечисляет деньги.

Инфраструктура для прохождения бизнес-процесса кредитной заявки:

1. Банковская система АБС Бисквит (СУБД Progress) - ведение банковских счетов
2. Барсик - Java-приложение, изначально разработанное для коллекторского отдела, но "обросшее" полезными функциями и поэтому использовавшееся повсеместно в банке.
3. Excel-форма для заполнения кредитной заявки и данных клиента.

Клиент посчитал, что сам процесс прохождения Кредитной заявки являлся плохо контролируемым и основывался на ручном контроле со стороны менеджеров.

Таким образом, целью проекта была замена устаревшей Excel-формы, а так же предоставление формального бизнес-процесса для прохождения Кредитной Заявки.

ООО «Артезио»
119333, Россия, Москва
Ул. Губкина, д. 3Г, офис 504

Тел.: +7 (495) 981-0531
Факс: +7 (495) 232-2683
Email: info@artezio.com
www.artezio.ru

Решение

База данных основана на схеме БД Барсика (PostgresQL), с добавлением к ней дополнительных таблиц. При необходимости модификации таблицы (например, добавление полей) для Барсика создавалось updatable view с исходным набором полей. Для отчетов была написана хранимая процедура.

Слой Бизнес-Логики: Java-приложение, общий каркас приложения выполнен с использованием Spring (Spring MVC, Spring Security), для доступа данных используется Hibernate. Бизнес-процесс реализован с использованием свободной библиотеки с открытыми исходными кодами Activity.

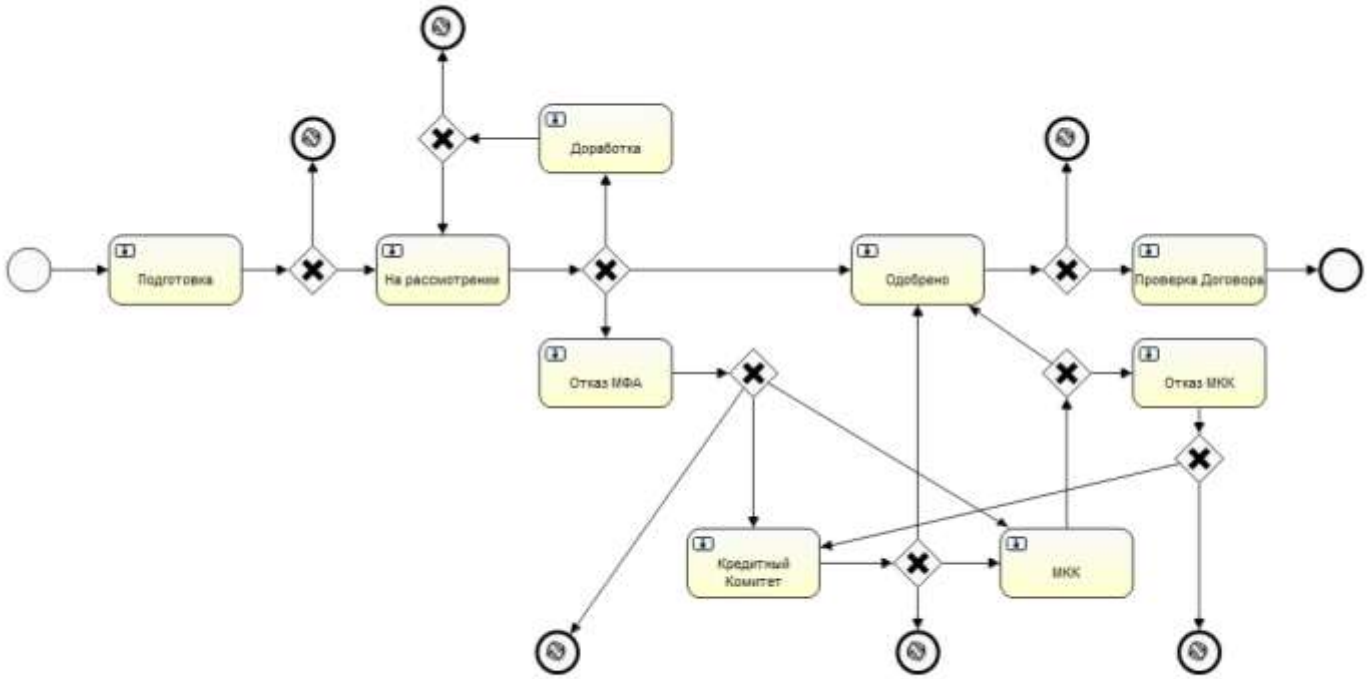
Представление данных: Spring MVC, JSP, JavaScript+Ajax(JQuery+plugins).

Для Бизнес-Логики использовался следующие слои:

- **DAO** - обеспечивал доступ к данным. Для каждой используемой сущности было написано своё DAO, например ClientDao { ... findByName(String name); ...}. Для транзакций, использовался подход OSIV - для каждой сессии открывалась отдельная транзакция. Это позволило избавиться от Fetch-запросов и связанных с этим лишних затрат времени на написание этих запросов.
- **Service** - Каждый сервис предоставлял удобный интерфейс к своей части задач, например, ProcessService - к библиотеке Бизнес-процесса, FileService - к хранилищу файлов и т.д.
- **Controller** - Получает данные от пользователя, вызывает необходимые сервисы, конвертирует данные из модели в доменные сущности и обратно когда это необходимо.

Для уровня представления использовались JSP совместно с JavaScript контролами. Данные на генерируемых HTML страницах подгружались как статически, при генерации JSP, так и динамически, с помощью ajax-запросов. В качестве Ajax-транспорта использовался Json.

Бизнес-процесс
Для пользователей системы был разработан фиксированный набор экранов, но в зависимости от роли пользователя и этапа кредитной заявки, происходила настройка каждого экрана – показ/скрытие полей, отключение возможности редактировать поле и т.д.



Тестирование

Заказчик не выдвигал требования об обязательном покрытии какого-то процента кода тестами, поэтому тесты были разработаны в основном для ключевых мест системы. Также приемочное и регрессионное тестирование планировалось проводить средствами заказчика по окончании проекта. Решение сопутствующих проблем и исправление замечаний также планируется силами заказчика.

Однако код структурирован таким образом, чтобы можно было легко добавить юнит-тесты для любого компонента системы без изменений в коде. В качестве тест-системы использовался JUnit и Spring Test.

Также разработаны автоматические тесты на Selenuim, имитирующие различные сценарии прохождения процесса. При необходимости их можно расширить, увеличив покрытие остальной части проекта.

Результат

Продукт был передан заказчику в феврале 2012г. Заказчик планирует в конце марта 2012г. начать опытную эксплуатацию системы, до этого Заказчик силами своего IT-отдела проводит тестирование, адаптацию к требованиям других отделов Банка и интеграцию системы в существующую инфраструктуру Банка.

При этом выбранная методология Agile позволила сократить расходы на планирование и разработку детального дизайна.