# Оглавление



# Техническое задание

Разработать форму регистрации, состоящую из трех шагов:

1) Фамилия, имя отчество, телефон, e-mail, пароль, повтор пароля.
2) Дата рождения, серия и номер паспорта.
3) Регион, город, улица.

Должна быть организована навигация между шагами, к 1 и 2 шагу можно возвратиться и изменять данные.

Поля телефон, e-mail, серия номер паспорта – уникальные, нельзя давать пройти соответствующий шаг, если такие значения уже есть в базе данных.

Пользователь может прервать выполнение регистрации на любом шаге кроме первого, и может затем позже вернуться, выполнить логин, и продолжить регистрацию с того шага на котором он закончил.

Например: пользователь заполнил 1й шаг, нажал «Далее» и затем закрыл браузер. Такой пользователь может войти в систему и должен увидеть 2й шаг формы регистрации.

Пользователь, который полностью завершил регистрацию, должен иметь возможность заходить в свой личный кабинет. Личный кабинет можно сделать заглушкой, с отображением только ФИО.

# Технологии

Проект логически разделен на две части:

- Back-end на Spring Data
- Web UI на Ext-JS 4

В качестве базы данных используется MySQL, аутентификация пользователей выполнена с применением Spring Security.

Физически back-end и UI хранятся в одном проекте, серверные компоненты в стандартной ветке /src/main/java, UI в каталоге /src/main/webapp/WEB-INF/ . Проект собирается и запускается с помощью maven.

## Реализация

Обмен данными между Web UI и back-end компонентой выполнен на базе REST сервисов. Каждый ответ сервера (за исключением declined запросов на авторизацию) представляет собой конверт, который кроме полезных *данных* содержит поля *статуса*, *кодов ошибок* и *код рекомендованной операции* для Web UI. В качестве полезных данных подразумеваются данные пользователя (например паспортные данные или данные по адресу проживания). В полях кодов ошибок передаются результаты проверки данных на стороне сервера, эти коды можно рассматривать как результат проверки полей простых объектов (User, IdCard, Place). Поле с кодом рекомендованной операции используется для перемещения клиента по формам регистрации, этот код отражает результат проверки на уровне группы сложных объектов. В дальнейшем код рекомендованной операции может использоваться бизнес логикой серверной компоненты в качестве сигнала в сторону UI для выполнения определенных операций, например для уведомлений пользователю или для подготовки новых данных для бизнес логики приложения.

Web UI представляет собой три простых модуля: страница авторизации и регистрации, пошаговый мастер регистрации и страница с заглушкой для кабинета пользователя. Все компоненты вылеплены на базе Фреймворка Ext-JS с применением MVC паттерна. Мастер регистрации реализует пошаговое перемещение вдоль форм на основе связанного списка (смотрите классы "controller.Registation" и "core.LinkedList"). При каждом шаге вперед и назад UI сохраняет данные формы на back-end сервер через REST сервисы (только если форма валидная, проверка на этом этапе выполняется только на основе формата и длинны данных в полях формы). Как было описано выше, сервер всегда возвращает стандартный конверт с ответом, в котором может присутствовать код операции, рекомендованный сервером. Связанный список, по которому пользователь перемещается по формам, не позволит перейти дальше формы, код которой находится за пределами значения, рекомендованного сервером. Кроме того, код операции рекомендованной сервером используется при повторном входе в систему при ситуации, в которой пользователь не закончил регистрацию на каком то шаге. Такой подход позволяет развязать логику работы (и соответственно реализации) UI и серверной компоненты. На стороне сервера выполняются сложные комплексные проверки, при этом по результатам таких проверок UI компонента вправе выбирать то или иное действие для того, чтобы обеспечить ввод данных способом, наиболее комфортным для конечного пользователя.

Серверная часть реализована в виде REST сервисов на основе Spring Data. Данные форм хранятся в трех сущностях: User, IdCard и Place. Для обработки каждой сущности созданы Data Repository, которые инкапсулируют сохранение и загрузку данных средствами библиотеки Spring Data. Физически данные хранятся в MySQL сервере. Авторизация пользователя реализована с применением библиотеки Spring Security.

Проверка данных на стороне сервера реализована на базе Hibernate Validator. В классической реализации Validator прозрачно проверяет сущности на предмет их корректного заполнения перед сохранением в "базу данных". В нашем случае такой подход не является оптимальным, так как, во первых, UI передает данные небольшими частями, при этом не всегда возможно сразу создать и сохранить полностью корректную сущность. Во вторых, сложно на лету обрабатывать ошибки, которые выбрасываются Hibernate слоем приложения и зачастую могут быть перехвачены и обработаны серверным Фреймворком за пределами нашего кода. По этим причинам нативная Hibernate валидация "на лету" отключена в конфигурации (смотрите параметр javax.persistence.validation.mode в файле application.properties), вместо этого серверная компонента создает локальный объект Hibernate Validator и выполняет проверку группы связанных сущностей (User, IdCard, Place) внутри кода, который находится под нашим контролем. В результате локальной проверки мы точно определяем некорректно заполненные сущности и поля, которые не удовлетворяют тем или иным требованиям. На основе этих данных мы добавляем коды ошибок и код рекомендованной операции в конверт, который отправляется в сторону UI.

## Тесты

Тесты Web UI не реализованы (а надо бы).

Покрытие серверных компонентов unit тестами составляет более 80%.

Интеграционные тесты отсутствуют.

## Запуск

Требования к системе: MySQL 5.6, JDK, Git, Maven, расово правильный браузер Chrome или Firefox.

Для запуска проекта необходимо установить MySQL сервер версии 5.6 (проект проверялся на версии 5.6.12.2 (community edition)). Запустите сервер и создайте пользователя cabinet с паролем cabinet. Можно использовать любые другие допустимые комбинации логина и пароля, в этом случае не забудьте указать их в конфигурационном файле application.properties проекта. Добавьте пользователю права, позволяющие создавать и удалять базы и таблицы.

Загрузите проект с GitHub:

```
git clone https://github.com/d70a/test.git
```

Перейдите в каталог проекта:

```
cd test
```

Запустите тестирование проекта (обязательно, в процессе тестирования будет создана база в MySQL):

```
mvn test
```

Запустите проект:

```
mvn clean tomcat7:run
```

Откройте адрес проекта в браузере:

http://localhost:8080/

## Плюшки

Локализация Web UI интерфейса, по умолчанию все поля и сообщения в JavaScript коде написаны на английском. Если в браузер пользователя установлена 'ru_XX'  локализация, то будет загружен русифицированный интерфейс.

Авторизация с использованием Spring Security.

Тесты с использованием Spring Security, анализ объема покрытия с помощью Cobertura.

Автоматическое создание схемы базы данных на лету.