# Техническое задание на программу для ЭВМ «Выбор элемента дерева» версия 1.

# 1. Описание

Программа представляет собой приложение, выбирающее элемент из дерева, представленного в виде:

* структуры в базе данных,
* структуры в файле (в формате json),
* структуры, переданной в API (в формате json),

на основе нескольких параметров:

* вес элемента,
* счетчик предыдущих выборов,
* число, сгенерированное генератором случайных чисел,
* либо любых других параметров, определенных в структуре элемента дерева и в формуле выбора.

Приложение состоит из следующих частей:

* сервер, реализующий логику работы с БД, предоставляющий веб-приложение и API. Поставляется в вариантах для Windows и Linux;
* web-приложение ;
* десктопное приложение.

Для всех частей приложения прилагаются программы установки (кроме web-приложения).

# 2. Требования

Исходное дерево может храниться в базе данных на сервере, в локальной базе данных, либо в файле.

Web-клиент работает только с сервером, десктопный клиент может работать автономно, с локальной базой данных (SQLite).

Формулы выбора элемента хранятся в базе данных, одна из формул - по умолчанию, устанавливается автоматически для нового дерева.

Должен быть предусмотрен механизм синхронизации локальной и серверной баз данных.

Авторизация не предусмотрена.

# 3. Логика работы

Пользователь работает либо в web-интерфейсе, либо в клиентском приложении. Логика работы в этих двух интерфейсах за небольшим исключением не отличается.

Основные действия пользователя:

* Создание дерева, редактирование (в том числе заполнение элементами), удаление.
* Добавление, редактирование, удаление формул получения следующего элемента
* Получение следующего элемента
* В интерфейсе десктопного клиента: разрешение конфликтов синхронизации, построение дерева из ФС
* Вызов API: вычисление следующего элемента без добавления дерева в БД (вызов принимает в качестве параметров дерево и формулу)

В системе существуют следующие требования:

* уникальность наименований элементов дерева, имеющих общего родителя (непосредственного)
* Уникальность наименований деревьев и формул
* формула должна соответствовать шаблону :
  1. AGR(Expression), где
* AGR – одна из агрегатных функций MaxIdFrom, MinIdFrom, AvgIdFrom\*, возвращающих Id элемента, значение выражения Expression которого макисмально, минимально или наиболее близко к среднему значению из переданного дерева элементов.
* Expression – выражение, возвращающее числовое значение, состоящее из математических операторов библиотеки Ncalc, чисел и полей элемента дерева (Weight, SelectCount, поля, указанные в поле AddFields)

\* AvgIdFrom ищет элемент соответствия начиная от среднего значения (округленного) и постепенно увеличивая погрешность по единице в обе стороны, пока не будет найден хотя бы один элемент

Дополнительно требуется добавить функции MaxFrom, MinFrom, AvgFrom, вычисляющие максимальное, минимальное и среднее значения соответственно всех элементов, переданных на вычисление.

В результате возвращается первый из элементов, удовлетворяющий выражению, либо null, если ни один из элементов не соответствует выражению.

# 4. Механизм синхронизации баз данных.

Синхронизация баз данных работает на стороне клиента в фоновом режиме, если режим работы приложения не установлен «ThickOnly». Логика работы сервиса синхронизации:

* При запуске клиента из таблицы конфигурации получается дата последней синхронизации.
* При каждой итерации сервиса с сервера вызывается список изменений по каждой из таблиц.
* Сохраняются изменения в локальную базу данных.
* Сохраняются записи несинхронизированных записей из локальной БД на серверную.
* В таблицу конфигурации записывается дата последней синхронизации.

# 5. Технологии

## 5.1. Общие компоненты

* Newtonsoft.Json (для работы с Json)
* NCalc (парсинг формул)
* .Net (Framework, Core)

## 5.2. Сервер

* EntityFramework
* ASP .NetCore
* .NetCore 3.x

## 5.3. Клиент

* SqLite (+ EntityFramework?)
* .Net Framework 4.7
* WPF
* HttpClient

## 5.4. Web-client

* ASP .NetCore
* ReactJS или AngularJS
* Html
* Css
* JQuery

## 5.5. Базы данных

* Сервер: PostgreSQL
* Клиент: SQLite

## 5.6. Тестирование

* XUnit
* Moq

## 5.7. Установщик

* WixInstaller

# 6. Версионирование

Исходные коды хранятся на Github. Очередной релиз выпускается после окончания всех работ по очередному ТЗ и прохождения всех тестов. Мажорная версия выпускается если доработки не содержат обратной совместимости. Иначе выпускается минорная версия. Для исправлений и мелких доработок выпускается патч.

Нумерация:

<Мажорная версия>.<минорная версия>.<Патч>

# 7. Лицензирование

Продукт распространяется по лицензии Apache 2.0.

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,

and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by

the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all

other entities that control, are controlled by, or are under common

control with that entity. For the purposes of this definition,

"control" means (i) the power, direct or indirect, to cause the

direction or management of such entity, whether by contract or

otherwise, or (ii) ownership of fifty percent (50%) or more of the

outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity

exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,

including but not limited to software source code, documentation

source, and configuration files.

"Object" form shall mean any form resulting from mechanical

transformation or translation of a Source form, including but

not limited to compiled object code, generated documentation,

and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or

Object form, made available under the License, as indicated by a

copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

form, that is based on (or derived from) the Work and for which the

editorial revisions, annotations, elaborations, or other modifications

represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain

separable from, or merely link (or bind by name) to the interfaces of,

the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including

the original version of the Work and any modifications or additions

to that Work or Derivative Works thereof, that is intentionally

submitted to Licensor for inclusion in the Work by the copyright owner

or by an individual or Legal Entity authorized to submit on behalf of

the copyright owner. For the purposes of this definition, "submitted"

means any form of electronic, verbal, or written communication sent

to the Licensor or its representatives, including but not limited to

communication on electronic mailing lists, source code control systems,

and issue tracking systems that are managed by, or on behalf of, the

Licensor for the purpose of discussing and improving the Work, but

excluding communication that is conspicuously marked or otherwise

designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity

on behalf of whom a Contribution has been received by Licensor and

subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable

copyright license to reproduce, prepare Derivative Works of,

publicly display, publicly perform, sublicense, and distribute the

Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(except as stated in this section) patent license to make, have made,

use, offer to sell, sell, import, and otherwise transfer the Work,

where such license applies only to those patent claims licensable

by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s)

with the Work to which such Contribution(s) was submitted. If You

institute patent litigation against any entity (including a

cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses

granted to You under this License for that Work shall terminate

as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the

Work or Derivative Works thereof in any medium, with or without

modifications, and in Source or Object form, provided that You

meet the following conditions:

(a) You must give any other recipients of the Work or

Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices

stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works

that You distribute, all copyright, patent, trademark, and

attribution notices from the Source form of the Work,

excluding those notices that do not pertain to any part of

the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its

distribution, then any Derivative Works that You distribute must

include a readable copy of the attribution notices contained

within such NOTICE file, excluding those notices that do not

pertain to any part of the Derivative Works, in at least one

of the following places: within a NOTICE text file distributed

as part of the Derivative Works; within the Source form or

documentation, if provided along with the Derivative Works; or,

within a display generated by the Derivative Works, if and

wherever such third-party notices normally appear. The contents

of the NOTICE file are for informational purposes only and

do not modify the License. You may add Your own attribution

notices within Derivative Works that You distribute, alongside

or as an addendum to the NOTICE text from the Work, provided

that such additional attribution notices cannot be construed

as modifying the License.

You may add Your own copyright statement to Your modifications and

may provide additional or different license terms and conditions

for use, reproduction, or distribution of Your modifications, or

for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with

the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,

any Contribution intentionally submitted for inclusion in the Work

by You to the Licensor shall be under the terms and conditions of

this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify

the terms of any separate license agreement you may have executed

with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade

names, trademarks, service marks, or product names of the Licensor,

except as required for reasonable and customary use in describing the

origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or

agreed to in writing, Licensor provides the Work (and each

Contributor provides its Contributions) on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or

implied, including, without limitation, any warranties or conditions

of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A

PARTICULAR PURPOSE. You are solely responsible for determining the

appropriateness of using or redistributing the Work and assume any

risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,

whether in tort (including negligence), contract, or otherwise,

unless required by applicable law (such as deliberate and grossly

negligent acts) or agreed to in writing, shall any Contributor be

liable to You for damages, including any direct, indirect, special,

incidental, or consequential damages of any character arising as a

result of this License or out of the use or inability to use the

Work (including but not limited to damages for loss of goodwill,

work stoppage, computer failure or malfunction, or any and all

other commercial damages or losses), even if such Contributor

has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing

the Work or Derivative Works thereof, You may choose to offer,

and charge a fee for, acceptance of support, warranty, indemnity,

or other liability obligations and/or rights consistent with this

License. However, in accepting such obligations, You may act only

on Your own behalf and on Your sole responsibility, not on behalf

of any other Contributor, and only if You agree to indemnify,

defend, and hold each Contributor harmless for any liability

incurred by, or claims asserted against, such Contributor by reason

of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

boilerplate notice, with the fields enclosed by brackets "[]"

replaced with your own identifying information. (Don't include

the brackets!) The text should be enclosed in the appropriate

comment syntax for the file format. We also recommend that a

file or class name and description of purpose be included on the

same "printed page" as the copyright notice for easier

identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

Наиболее современная и сбалансированная из разрешительных лицензий. Написана человеческим языком, но с оглядкой на современное правоприменение, в частности, упомянутые выше патентные отношения (пункт 3 лицензии). GNU советуют применять именно эту лицензию, когда вам необходима разрешительная лицензия.

Для применения лицензии Apache 2.0 к вашему проекту, нужно добавить в него файл LICENSE, содержащий текст лицензии. Кроме того, в APPENDIX лицензии нам предлагают добавлять в качестве шапки в каждый файл исходного кода следующий текст:

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

Но при этом сама лицензия выдвигает следующие требования:

made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below)

copyright notice — это как раз строка, указывающая правообладателя. А «made available under the License, as indicated» означает, что еще должна быть явно указана лицензия. То есть, допустимо что-то вида:

//Copyright 2021 Dmitriy Rokoth

//Licensed under the Apache License, Version 2.0

Причем, совсем необязательно в исходном коде — Apache 2.0 позволяет для этого использовать файл NOTICE («or attached to the work»).

И еще о файле NOTICE: если в вашей работе вы используете чужой проект под лицензией Apache 2.0, содержащий свой файл NOTICE, то в этом случае вы обязаны копировать в производную работу содержимое файла NOTICE, в одно из трех мест: либо в аналогичный файл NOTICE, либо в исходные коды или документацию, распространяемую вместе с производной работой, либо в вывод производной работы (например в about-диалог); все согласно пункту 4 (d) лицензии. Заметьте, что, вопреки расхожему мнению, обязательного наличия файла NOTICE лицензия не требует.

При распространении в бинарной форме, вы, кроме того, должны предоставлять копию лицензии вместе с программой.

# 8. Модели данных

## 8.1. Клиентские модели

### 8.1.1. Базовый класс

Базовый класс Entity содержит общие поля (свойства) для всех моделей данных

#### 8.1.1.1. Entity

***Entity***

***{***

***public Guid Id;***

***}***

#### 8.1.1.1. EntityHistory

***EntityHistory***

***{***

***public long HId;***

***public DateTime ChangeDate;***

***}***

### 8.1.2. Дерево

#### 8.1.2.1. Описание

***Tree: Entity***

***{***

***public string Name;***

***public string Description;***

***public Guid FormulaId;***

***}***

#### 8.1.2.2. Элементы дерева

***TreeItem: Entity***

***{***

***public string Name;***

***public Guid TreeId;***

***public Guid? ParentId;***

***public bool IsLeaf;***

***public int SelectCount;***

***public int Weight;***

***public json AddFields;***

***}***

#### 8.1.2.3. История изменений дерева

***TreeHistory: EntityHistory***

***{***

***public string Name;***

***public string Description;***

***public Guid FormulaId;***

***}***

#### 8.1.2.4. История изменений элементов дерева

***TreeItemHistory: EntityHistory***

***{***

***public string Name;***

***public Guid TreeId;***

***public Guid? ParentId;***

***public bool IsLeaf;***

***public int SelectCount;***

***public int Weight;***

***public json AddFields;***

***}***

#### 8.1.2.5. Создание дерева

***TreeCreator***

***{***

***public string Name;***

***public string Description;***

***public Guid FormulaId;***

***}***

#### 8.1.2.6. Обновление дерева

***TreeUpdater***

***{***

***public Guid Id;***

***public string Name;***

***public string Description;***

***public Guid FormulaId;***

***}***

#### 8.1.2.6. Изменение элементов дерева

***TreeItemsUpdater***

***{***

***public Guid? Id ;***

***public Guid TreeId;***

***public string Name;***

***public int Weight;***

***public string AddFields;***

***public IEnumerable<TreeItemUpdater> TreeItems;***

***}***

#### 8.1.2.5. Создание элемента дерева

***TreeItemCreator***

***{***

***public Guid TreeId;***

***public Guid? ParentId;***

***public string Name;***

***public int Weight;***

***public string AddFields;***

***}***

#### 8.1.2.5. Обновление элемента дерева

***TreeItemUpdater***

***{***

***public Guid Id;***

***public Guid TreeId;***

***public Guid? ParentId;***

***public string Name;***

***public int Weight;***

***public string AddFields;***

***}***

### 8.1.3. Формула

#### 8.1.3.1. Formula

***Formula: Entity***

***{***

***public string Name;***

***public string Text;***

***public bool IsDefault;***

***}***

#### 8.1.3.2. FormulaCreator

***FormulaCreator***

***{***

***public string Name;***

***public string Text;***

***public bool IsDefault;***

***}***

#### 8.1.3.3. FormulaUpdater

***FormulaUpdater***

***{***

***public Guid Id;***

***public string Name;***

***public string Text;***

***public bool IsDefault;***

***}***

#### 8.1.3.4. История изменений формул

***Formula: EntityHistory***

***{***

***public string Name;***

***public string Text;***

***public bool IsDefault;***

***}***

### 8.1.4. Выборка элемента

#### 8.1.4.1. Запрос выборки элемента

***SelectRequest***

***{***

***public Guid? TreeId;***

***public int Count = 1;***

***public bool LeafOnly = true;***

***}***

#### 8.1.4.2. Запрос выборки элемента дерева (дерево в запросе)

***SelectCustomRequest***

***{***

***public IEnumerable<TreeItem> Tree;***

***public string Formula;***

***public int Count = 1;***

***public bool LeafOnly = true;***

***}***

#### 8.1.4.3. Результат выборки элемента

***SelectResponse***

***{***

***public IEnumerable<SelectResponseElement> Result;***

***}***

***SelectResponseElement***

***{***

***public Guid Id;***

***public string Name;***

***public string FullPath;***

***}***

### 8.1.5. Фильтры

#### 8.1.5.1. Базовый класс

***Filter {***

***public int Page;***

***public int Size;***

***public string Sort;***

***public bool WithExtension;***

***}***

#### 8.1.5.2. TreeFilter

***TreeFilter: Filter {***

***public string Name;***

***}***

#### 8.1.5.3. FormulaFilter

***FormulaFilter: Filter {***

***public string Name;***

***}***

#### 8.1.5.4. TreeItemFilter

***TreeItemFilter: Filter {***

***public string Name;***

***public Guid TreeId;***

***}***

#### 8.1.5.5. TreeItemHistoryFilter

***TreeItemHistoryFilter: Filter {***

***public string Name;***

***public Guid TreeId;***

***public DateTimeOffset? From;***

***public DateTimeOffset? To;***

***public Guid Id;***

***}***

#### 8.1.5.6. TreeHistoryFilter

***TreeHistoryFilter: Filter {***

***public string Name;***

***public DateTimeOffset? From;***

***public DateTimeOffset? To;***

***public Guid Id;***

***}***

#### 8.1.5.7. FormulaHistoryFilter

***FormulaHistoryFilter: Filter {***

***public string Name;***

***public DateTimeOffset? From;***

***public DateTimeOffset? To;***

***public Guid Id;***

***}***

## 8.2. Серверные (внутренние) модели

### 8.2.1 Базовый класс

Базовый класс Entity содержит общие поля (свойства) для всех моделей данных

#### 8.2.1.1. Entity

***Entity***

***{***

***public Guid Id;***

***public DateTime VersionDate;***

***public bool IsDeleted;***

***}***

#### 8.2.1.2. EntityHistory

***EntityHistory: Entity***

***{***

***public DateTime ChangeDate;***

***public Guid HId;***

***}***

# Not actualized

### 8.2.2. Дерево

#### 8.2.2.1. Описание

***Tree: Entity***

***{***

***public string Name;***

***public string Description;***

***public Guid FormulaId;***

***}***

#### 8.2.2.2. Элементы дерева

***TreeItem: Entity***

***{***

***public string Name;***

***public Guid TreeId;***

***public Guid? ParentId;***

***public bool IsLeaf;***

***public int SelectCount;***

***public int Weight;***

***public json AddFields;***

***}***

#### 8.2.2.3. История изменений дерева

***TreeHistory: EntityHistory***

***{***

***public string Name;***

***public string Description;***

***public Guid FormulaId;***

***}***

#### 8.2.2.4. История изменений элементов дерева

***TreeItemHistory: EntityHistory***

***{***

***public string Name;***

***public Guid TreeId;***

***public Guid? ParentId;***

***public bool IsLeaf;***

***public int SelectCount;***

***public int Weight;***

***public json AddFields;***

***}***

### 8.2.3. Формулы

***Formula: Entity***

***{***

***public string Name;***

***public string Text;***

***public bool IsDefault;***

***}***

***FormulaHistory: EntityHistory***

***{***

***public string Name;***

***public string Text;***

***public bool IsDefault;***

***}***

## 8.3. Модели базы данных

### 8.3.1. Сервер

#### 8.3.1.1. tree

tree:

*id uuid not null primary key,*

*name varchar(100) not null,*

*description varchar(1000),*

*formula\_id uuid not null,*

*version\_id uuid not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null*

*constraints:*

*foreign key formula\_id on formula*

*unique name where not is\_deleted*

*unique version\_id*

*unique (version\_num, id)*

#### 8.3.1.2. tree\_item

tree\_item:

*id uuid not null primary key,*

*name varchar(100) not null,*

*description varchar(1000),*

*tree\_id uuid not null,*

*parent\_id uuid null,*

*is\_leaf boolean not null,*

*select\_count int not null default 0,*

*weight int not null default 1,*

*add\_fields jsonb null,*

*version\_id uuid not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null*

*constraints:*

*foreign key tree\_id on tree,*

*foreign key parent\_id on tree\_item,*

*unique (name, parent\_id) where not is\_deleted,*

*unique version\_id,*

*unique (version\_num, id)*

#### 8.3.1.3. formula

formula:

*id uuid not null primary key,*

*name varchar(100) not null,*

*text varchar(1000),*

*is\_default Boolean not null,*

*version\_id uuid not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null*

*constraints:*

*unique name where not is\_deleted*

*unique version\_id*

*unique (version\_num, id )*

#### 8.3.1.4. h\_tree

h\_tree:

*h\_id uuid not null primary key,*

*id uuid not null,*

*name varchar(100),*

*description varchar(1000),*

*formula\_id uuid,*

*version\_id uuid,*

*version\_num int,*

*version\_date timestamptz,*

*is\_deleted boolean,*

*change\_date timestamptz not null,*

#### 8.3.1.5. h\_tree\_item

h\_tree\_item:

*h\_id uuid not null primary key,*

*id uuid,*

*name varchar(100),*

*description varchar(1000),*

*tree\_id uuid,*

*parent\_id uuid,*

*is\_leaf boolean,*

*select\_count int,*

*weight int,*

*add\_fields jsonb,*

*version\_id uuid,*

*version\_num int,*

*version\_date timestamptz,*

*is\_deleted boolean,*

*change\_date timestamptz not null*

#### 8.3.1.6. h\_formula

h\_formula:

*h\_id uuid not null primary key,*

*id uuid not null primary key,*

*name varchar(100) not null,*

*text varchar(1000),*

*is\_default Boolean not null,*

*version\_id uuid not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null*

*change\_date timestamptz not null*

#### 8.3.1.7. history\_trigger

Триггер срабатывает при любом изменении данных в таблице и записывает старое значение строки в историческую таблицу.

#### 8.3.1.8. delete\_allow\_trigger

Триггер срабатывает при попытке удаления записи, и, если не выставлен флаг, разрешающий удаление, генерирует исключение.

#### 8.3.1.9. settings

settings:

*id int not null,*

*param\_name varchar not null,*

*param\_value varchar null,*

### 8.3.2. Клиент

#### 8.3.2.1. tree

tree:

*id varchar not null primary key,*

*name varchar(100) not null,*

*description varchar(1000),*

*formula\_id varchar not null,*

*version\_id varchar not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null,*

#### 8.3.2.2. tree\_item

tree\_item:

*id varchar not null primary key,*

*name varchar(100) not null,*

*description varchar(1000),*

*tree\_id varchar not null,*

*parent\_id varchar null,*

*is\_leaf boolean not null,*

*select\_count int not null default 0,*

*weight int not null default 1,*

*add\_fields varchar null,*

*version\_id varchar not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null*

#### 8.3.2.3. formula

formula:

*id varchar not null primary key,*

*name varchar(100) not null,*

*text varchar(1000),*

*is\_default Boolean not null,*

*version\_id varchar not null,*

*version\_num int not null,*

*version\_date timestamptz not null,*

*is\_deleted boolean not null*

#### 8.3.2.4. sync\_conflict

sync\_conflict:

*s\_id varchar not null primary key,*

*id varchar not null,*

*table varchar not null,*

*server\_version\_date timestamptz not null,*

*server\_is\_deleted boolean not null,*

*local\_version\_date timestamptz not null,*

*local\_is\_deleted boolean not null*

#### 8.3.2.5. settings

settings:

*id int not null,*

*param\_name varchar not null,*

*param\_value varchar null,*

# 9. Сервер

## 9.1. Описание

Сервер представляет собой web-сервис, выполняющий функции работы с базой данных через ORM (Entity Framework). В первой версии программы реализуется коннектор только к PostgreSql.

Бизнес-логика приложения вынесена в компонент, доступный всем частям приложения для разных кейсов работы приложения.

Для работы с внешними клиентами сервер предоставляет API, хост которого располагается в той же сборке, что и Web клиент.

Сервер написан на C# под .Net Core 3.x.

## 9.2. Работа с базой данных

Работа с базой данных сделана через обертку (Repository).

Реализуется паттерн SoftDeleted (поле IsDeleted).

Поддерживается версионность записей для синхронизации с клиентом.

Реализуются следующие общие методы:

### 9.2.1. Методы работы с данными

#### 9.2.1.1. Get

Получение списка сущностей Entity согласно заданного фильтра. Фильтр содержит значения полей для фильтрации, пейджинг, сортировку, метод обогащения. Сигнатура:

***public Task<IEnumerable<Entity>> GetAsync(Filter<Entity> filter, CancellationToken token)***

#### 9.2.1.2. GetItem

Получение сущности Entity по Id с обогащением (с использованием переданного метода). В случае отсутствия сущности в базе возвращается null. Сигнатура:

***public Task<Entity> GetItemAsync(Guid id, Action<Entity> addAction, CancellationToken token)***

#### 9.2.1.3. Add

Добавление сущности Entity c генерированием Id по переданным полям. Если список полей для генерирования Id пустой или null, генерируется случайный Id. При добавлении используется оптимистичная блокировка по Id. Сигнатура:

***public Task<Entity> AddAsync(Entity entity, string[] keyFields, CancellationToken token)***

#### 9.2.1.4. Update

Обновление сущности Entity. При обновлении используется оптимистичная блокировка по полям Version… . Сигнатура:

***public Task<Entity> UpdateAsync(Entity entity, CancellationToken token)***

#### 9.2.1.5. Delete

Удаление сущности Entity по Id. Используется паттерн SoftDeleted. Сигнатура:

***public Task<Entity> DeleteAsync(Guid id, CancellationToken token)***

## 9.3. API

### 9.3.1. Дерево

#### 9.3.1.1. Метод получения списка деревьев

Метод принимает следующие параметры:

* Фильтр по имени (поиск по вхождению)
* Номер страницы
* Число элементов
* Поле сортировки

Возвращает список деревьев (Tree – клиентская модель), соответствующих фильтрам, в формате json. Сигнатура:

***GET /api/v1/tree/?name=name&page=1&size=10&sort=name***

#### 9.3.1.2. Метод получения дерева по Id

Метод принимает следующие параметры:

* id

Возвращает дерево (Tree – клиентская модель) в формате json. Сигнатура:

***GET /api/v1/tree/{id}***

#### 9.3.1.3. Метод получения всех элементов дерева

Метод принимает следующие параметры:

* id

Возвращает список элементов дерева (TreeItem – клиентская модель) в формате json. Сигнатура:

***GET /api/v1/tree/{id}/items***

#### 9.3.1.4. Метод добавления дерева

Метод принимает следующие параметры:

* TreeCreator

Возвращает добавленное дерево (Tree – клиентская модель) в формате json. Сигнатура:

***POST /api/v1/tree/***

#### 9.3.1.5. Метод изменения дерева

Метод принимает следующие параметры:

* TreeUpdater

Возвращает измененное дерево (Tree – клиентская модель) в формате json. Сигнатура:

***PUT /api/v1/tree/***

#### 9.3.1.6. Метод изменения элементов дерева

Метод принимает следующие параметры:

* TreeItemsUpdater

Возвращает измененный список (TreeItem – клиентская модель) в формате json. Сигнатура:

***PUT /api/v1/tree/{id}/items***

#### 9.3.1.7. Метод удаления дерева

Метод принимает следующие параметры:

* id

Возвращает удаленное дерево (Tree – клиентская модель) в формате json. Элементы дерева удаляются так же. Метод восстановления данных в текущей версии не предусмотрен. Сигнатура:

***DELETE /api/v1/tree/{id}***

#### 9.3.1.8. Метод получения списка изменений дерева

Метод принимает следующие параметры:

* id
* Фильтр по имени (поиск по вхождению)
* Дата изменения от
* Дата изменения до
* Номер страницы
* Число элементов
* Поле сортировки

Возвращает список изменений деревьев (TreeHistory – клиентская модель), соответствующих фильтрам, в формате json. Сигнатура:

***GET /api/v1/tree/changes/?id=id&name=name&page=1&size=10&sort=name***

#### 9.3.1.9. Метод получения списка изменений элементов дерева

Метод принимает следующие параметры:

* Id (дерева)
* Фильтр по имени дерева (поиск по вхождению)
* Фильтр по имени элемента (поиск по вхождению)
* Дата изменения от
* Дата изменения до
* Номер страницы
* Число элементов
* Поле сортировки

Возвращает список изменений элементов дерева (TreeItemHistory – клиентская модель), соответствующих фильтрам, в формате json. Сигнатура:

***GET /api/v1/tree/items/changes/?id=id&treename=treename&name=name&page=1&size=10&sort=name***

### 9.3.2. Формулы

#### 9.3.2.1. Метод получения списка формул

Метод принимает следующие параметры:

* Фильтр по имени (поиск по вхождению)
* Номер страницы
* Число элементов
* Поле сортировки

Возвращает список формул (Formula – клиентская модель), соответствующих фильтрам, в формате json. Сигнатура:

***GET /api/v1/formula/?name=name&page=1&size=10&sort=name***

#### 9.3.2.2. Метод получения формулы по Id

Метод принимает следующие параметры:

* id

Возвращает формулу (Formula – клиентская модель) в формате json. Сигнатура:

***GET /api/v1/ formula /{id}***

#### 9.3.2.3. Метод добавления формулы

Метод принимает следующие параметры:

* FormulaCreator (клиентская модель)

Возвращает добавленную формулу (Formula – клиентская модель) в формате json. Сигнатура:

***POST /api/v1/ formula /***

#### 9.3.2.4. Метод изменения формулы

Метод принимает следующие параметры:

* FormulaUpdater

Возвращает измененную формулу (Formula – клиентская модель) в формате json. Сигнатура:

***PUT /api/v1/ formula /***

#### 9.3.2.5. Метод удаления формулы

Метод принимает следующие параметры:

* id

Возвращает удаленную формулу (Formula – клиентская модель) в формате json. Сигнатура:

***DELETE /api/v1/ formula /{id}***

#### 9.3.2.6. Метод получения списка изменений

Метод принимает следующие параметры:

* id
* Фильтр по имени (поиск по вхождению)
* Дата изменения от
* Дата изменения до
* Номер страницы
* Число элементов
* Поле сортировки

Возвращает список изменений формул (FormulaHistory – клиентская модель), соответствующих фильтрам, в формате json. Сигнатура:

***GET /api/v1/formula/changes/?id=id&name=name&page=1&size=10&sort=name***

### 9.3.3. Выборка элемента из дерева в БД

Метод принимает следующие параметры:

* SelectRequest (клиентская модель)

Возвращает модель SelectResponse. Сигнатура:

***POST /api/v1/select/common/***

### 9.3.4. Выборка элемента из дерева в запросе

Метод принимает следующие параметры:

* SelectCustomRequest (клиентская модель)

Возвращает модель SelectResponse. Сигнатура:

***POST /api/v1/select/custom/***

### 9.3.5. Раскатка базы данных

Метод не принимает параметров. Выполняет раскатку серверной базы из скриптов.

Возвращает true/false. Сигнатура:

***POST /api/v1/deploy/***

### 9.3.6. Доступность сервера

Метод не принимает параметров. Возвращает OK если доступен. Сигнатура:

***GET /api/v1/ping/***

## 9.4. Сервис данных (прослойка между контроллером и репозиториями, общий компонент для API и web-клиента)

### 9.4.1. Метод GetTrees

Принимает параметром упакованные параметры запроса из контроллера TreeFilter, вызывает репозиторий Tree, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.2. Метод GetFormulas

Принимает параметром упакованные параметры запроса из контроллера FormulaFilter, вызывает репозиторий Formula, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.3. Метод GetTree

Принимает параметром идентификатор искомого дерева, вызывает репозиторий Tree, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.4. Метод GetTreeItem

Принимает параметром идентификатор искомого элемента дерева, вызывает репозиторий TreeItem, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.5. Метод GetFormula

Принимает параметром идентификатор искомой формулы, вызывает репозиторий Formula, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.6. Метод GetTreeItems

Принимает параметром id дерева, вызывает репозиторий TreeItem, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.7. Метод AddTree

Принимает параметром модель TreeCreator, вызывает метод Add репозитория Tree, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.8. Метод UpdateTreeItems

Принимает параметром модель TreeItemsUpdater, проверяет все ограничения, сравнивает новое дерево со старым, выполняет добавление новых элементов, изменение существующих и удаление отсутствующих в запросе, для каждого из элементов вызывает соотвествующий запрос репозитория TreeItem, возвращает новое дерево элементов.

### 9.4.9. Метод AddFormula

Принимает параметром модель FormulaCreator, вызывает метод Add репозитория Formula, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.10. Метод UpdateTree

Принимает параметром модель TreeUpdater, вызывает метод Update репозитория Tree, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.11. Метод UpdateFormula

Принимает параметром модель FormulaUpdater, вызывает метод Update репозитория Formula, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.12. Метод UpdateTreeItem

Принимает параметром модель TreeItemUpdater, вызывает метод Update репозитория TreeItem, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.13. Метод DeleteTree

Принимает параметром Id дерева, вызывает метод Remove репозитория Tree, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.13. Метод DeleteTreeItem

Принимает параметром Id элемента дерева, вызывает метод Remove репозитория TreeItem, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.13. Метод DeleteFormula

Принимает параметром Id формулы, вызывает метод Remove репозитория Formula, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.13. Метод Calculate

Принимает параметром модель SelectRequest, запрашивает элементы дерева, формулу, и вызывает метод Calculate библиотеки формул, формирует результат, сохраняет изменения в БД, возвращает модель SelectResponse.

### 9.4.13. Метод CalculateCustom

Принимает параметром модель SelectRequestCustom, подготавливает запрос, и вызывает метод Calculate библиотеки формул, формирует результат, возвращает модель SelectResponse.

### 9.4.14. Метод GetTreeHistory

Принимает параметром упакованные параметры запроса из контроллера TreeHistoryFilter, вызывает репозиторий TreeHistory, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.15. Метод GetTreeItemHistory

Принимает параметром упакованные параметры запроса из контроллера TreeItemHistoryFilter, вызывает репозиторий TreeItemHistory, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

### 9.4.16. Метод GetFormulaHistory

Принимает параметром упакованные параметры запроса из контроллера FormulaHistoryFilter, вызывает репозиторий FormulaHistory, маппит результат в клиентскую модель, выполняет дообогащение при необходимости.

## 9.5. Дополнительные модули

### 9.5.1. Деплоер БД

Модуль, использующий внешнюю библиотеку раскатки БД. Создает базу данных (скрипты в папке Deploy) в случае отсутствия (скрипт проверки наличия базы в папке Check), накатывает обновления (каталог Update).

## 9.6. Web-приложение

см. Web клиент

# 10. Библиотека расчета по формуле

В первой версии библиотека реализует один метод:

***Public IEnumerable<CalcResult> Execute (CalcRequest request)***

Где

***CalcResult***

***{***

***Public Guid Id;***

***}***

***CalcRequest***

***{***

***Public IEnumerable<CalcRequestItem> Items;***

***Public string Formula;***

***Public int Count;***

***}***

***CalcRequestItem***

***{***

***Public Guid Id;***

***Public string(json) Fields;***

***}***

Компонент вычисляет результат для всех элементов Items в CalcRequest до тех пор, пока не удастся получить результат (для некоторых случаев, н-р, расчет агрегирующих функций min, max etc, формулы всегда вычисляется для всех элементов). Если ни один из элементов не удовлетворяет условиям формулы, возвращается null. В случае ошибки в формуле генерируется ошибка.

В первой версии компонента используется библиотека NCalc.

# 11. Клиент

## 11.1. Главное окно

### 11.1.1. Меню

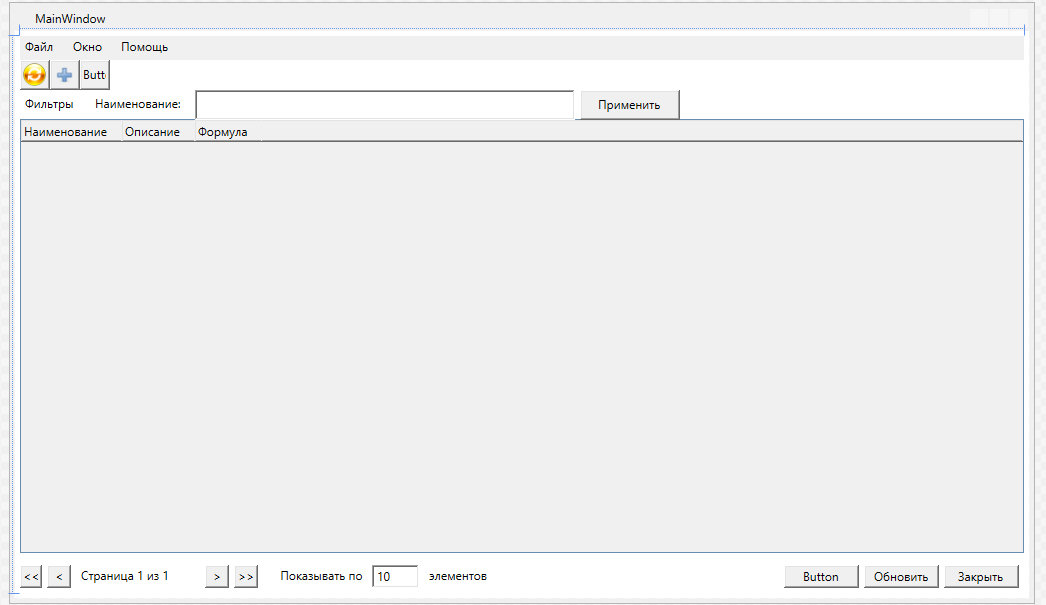
* Файл
  + Подключиться к серверу… (подключение к серверу)
  + Конфликты синхронизации … (окно конфликтов синхронизации)
  + ----
  + Добавить дерево (добавить дерево вручную)
  + Импорт из файла… (импорт дерева из файла)
  + Экспортировать в файл… (сохранение выбранных деревьев в файлы)
  + ----
  + Выход (закрытие программы)
* Окно
  + Дерево
  + Формулы
* Помощь
  + О программе
  + Сообщение об ошибке (сообщение на почту, позднее – сервис сбора ошибок)

### 11.1.2. Управляющие элементы

Управляющие элементы представлены в виде панели инструментов, частично дублирующих меню:

* Обновить (обновить таблицу)
* Добавить дерево/формулу
* Импорт из файла… (импорт дерева из файла, только для окна деревьев)
* Редактировать выбранное дерево/формулу
* Удалить выбранное дерево/формулу
* Переключиться на окно редактирования деревьев/формул
* Получить следующий элемент выбранного дерева (только для окна деревьев)
* Выход

### 11.1.3. Дизайн



### 11.1.4. Таблица

Табличный список деревьев с постраничной отдачей. Колонки таблицы:

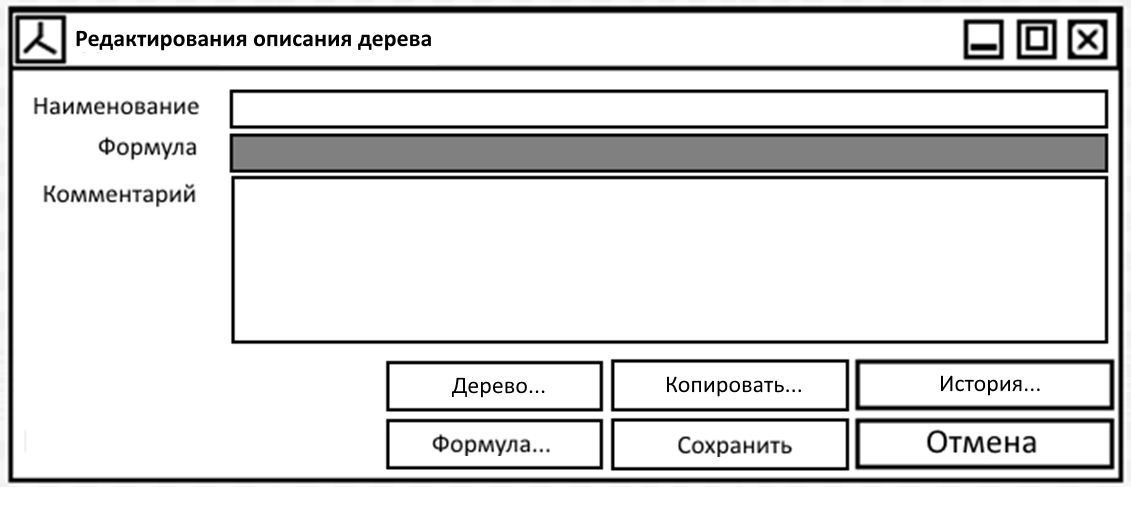
* идентификатор – скрытое поле
* наименование (Наименование)
* описание
* управляющие элементы (кнопки):
  + редактировать описание. Открывается страница редактирования описания дерева
  + редактировать дерево. Открывается страница редактирования дерева
  + изменить формулу выбора для данного дерева. Открывается окно выбора формулы
  + получить следующий элемент дерева. Открывается модальное окно элемента
  + удалить дерево. Открывается диалоговое окно с подтверждением удаления

## 11.2. Окно редактирования описания дерева

### 11.2.1. Управляющие элементы

* Кнопка «Сохранить»
* Кнопка «Копировать в новое дерево»
* Кнопка «Изменить формулу»
* Кнопка «Просмотреть историю изменений»
* Кнопка «Просмотр/Редактирование дерева»
* Кнопка «Удалить»
* Кнопка «Отменить»

### 11.2.2. Дизайн



### 11.2.3. Поля

* Наименование
* Комментарий
* Формула (readonly)

# Not formatted

* + 1. Логика.

Окно содержит два редактируемых поля, поле формула задается выбором нажатием на кнопку «Формула».

По кнопке «История» открывается окно истории изменений дерева.

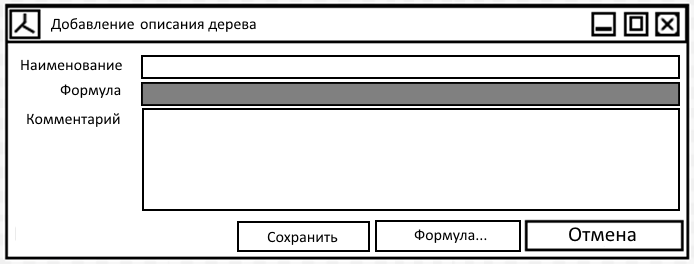
По кнопке «Удалить» открывается окно подтверждения удаления дерева.

По кнопке «Дерево» открывается окно редактирования дерева.

По кнопке «Копировать» открывается окно добавления описания дерева (в случае сохранения дерево будет скопировано в новое).

По кнопке «Сохранить» описание дерева сохраняется в базу. По кнопке «Отмена» форма закрывается без сохранения.

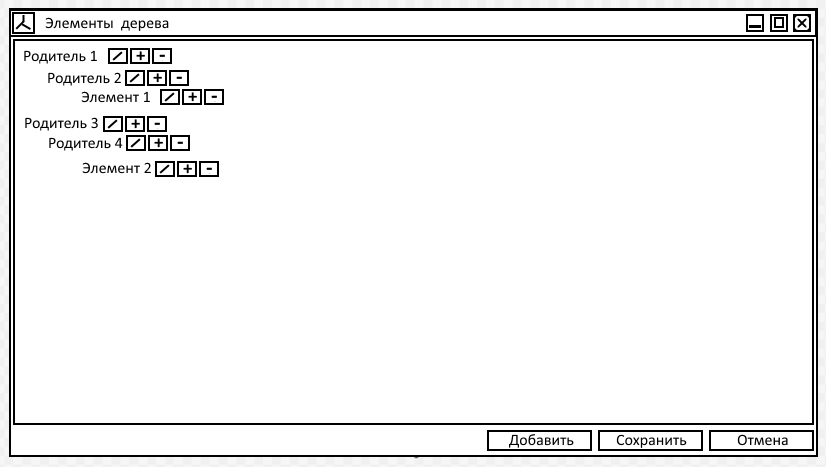
* 1. Окно добавления описания дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Выбрать формулу»
* Кнопка «Отменить»
  + 1. Дизайн



* + 1. Поля
* Текстовое поле: Наименование
* Текстовое поле: Комментарий
* Текстовое поле: Формула (readonly)
  + 1. Логика

Окно предназначено для добавления нового дерева (описания). Содержит поля «Наименование», «Формула» (readonly), «Комментарий». Формулу можно выбрать по кнопке «Формула…». По кнопке «Сохранить» модель сохраняется в базу данных, в случае успеха открывается окно редактирования описания дерева. По кнопке «Отмена» окно закрывается без сохранения.

* 1. Окно редактирования элементов дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Отменить»
* Кнопка «Добавить корневой элемент»
* Управляющие элементы ветвей дерева:
  + Добавить (дочерний элемент)
  + Удалить (удаляется вместе с дочерними)
  + Редактировать (открывается форма редактирования элемента)
    1. Дизайн



* + 1. Поля

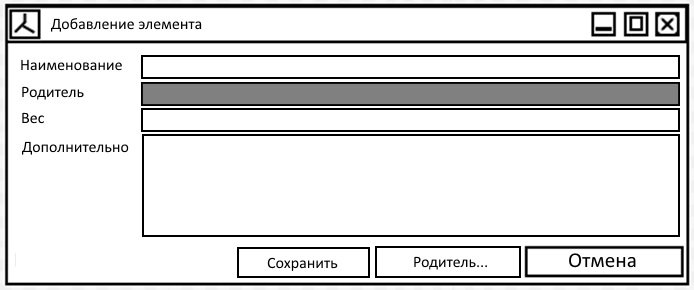
- Tree-компонент: дерево элементов

* + 1. Логика

Окно отображает выбранное дерево в виде иерархической структуры. По нажатию на кнопку редактирования открывается окно редактирования выбранного элемента. По нажатию на кнопку добавления открывается окно добавления элемента с родителем, раным выбранному элементу. По нажатию на кнопку удаления элемент и все дочерние элементы удаляются.

Кнопка «Добавить» добавляет корневой элемент. По нажатию на кнопку «Сохранить» дерево сохраняется в базу данных. По нажатию на кнопку «Отмена» окно закрывается без сохранения.

* 1. Окно добавления элемента дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Отменить»
  + 1. Дизайн



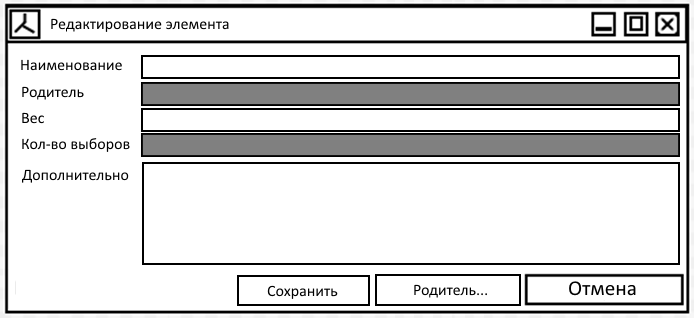
* + 1. Поля
* Наименование
* Родитель (readonly)
* Вес
* Дополнительно (textarea, для хранения дополнительных полей)
  + 1. Логика

Поля «родитель» и «кол-во» выборов readonly (родителя можно изменить по кнопке «Изменить родителя»). Наименование должно быть уникальным для всех дочерних элементов родителя. В поле «Дополнительно» данные хранятся в формате json. Вес - целое положительное число (больше 0).

По кнопке «Сохранить» данные сохраняются в структуру дерева (без сохранения в БД), форма закрывается.

По кнопке «Отменить» изменения не сохраняются, форма закрывается (с подтверждением).

* 1. Окно редактирования элемента дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Сменить родителя»
* Кнопка «Отменить»
  + 1. Дизайн



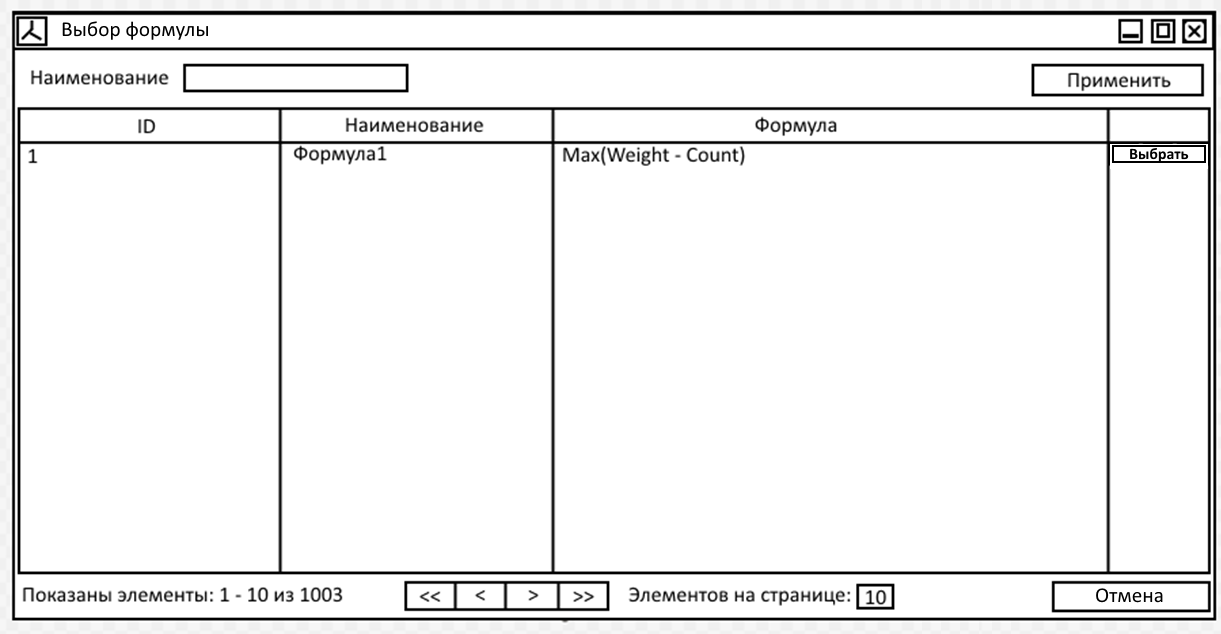
* + 1. Поля
* Наименование
* Родитель (readonly)
* Кол-во выборов (readonly)
* Вес
* Дополнительно (textarea, для хранения дополнительных полей)
  + 1. Логика

Поля «родитель» и «кол-во выборов» - readonly (родителя можно изменить по кнопке «Изменить родителя»). Наименование должно быть уникальным для всех дочерних элементов родителя. В поле «Дополнительно» данные хранятся в формате json. Вес - целое положительное число (больше 0).

По кнопке «Сохранить» данные сохраняются в структуру дерева (без сохранения в БД), форма закрывается.

По кнопке «Отменить» изменения не сохраняются, форма закрывается (с подтверждением).

* 1. Окно выбора формулы
     1. Управляющие элементы
* Кнопка «Выбрать» напротив каждой формулы
* Кнопка «Применить» в блоке фильтров
* Кнопка «Отмена»
* Элементы пагинатора
  + 1. Дизайн



* + 1. Поля
* Таблица: список формул
* Фильтры:
  + Текстовое поле: «Наименование»
* Пагинатор:
  + Числовое поле: кол-во элементов на странице
    1. Логика

Форма предназначена для выбора формулы, вызывается с формы редактирования описания дерева. По нажатия кнопки «Выбрать» возвращает в вызывающую форму Id выбранной формулы.

* 1. Окно списка формул
     1. Управляющие элементы

- Кнопка «Добавить»

- Напротив каждой формулы кнопки:

- Редактировать

- Удалить

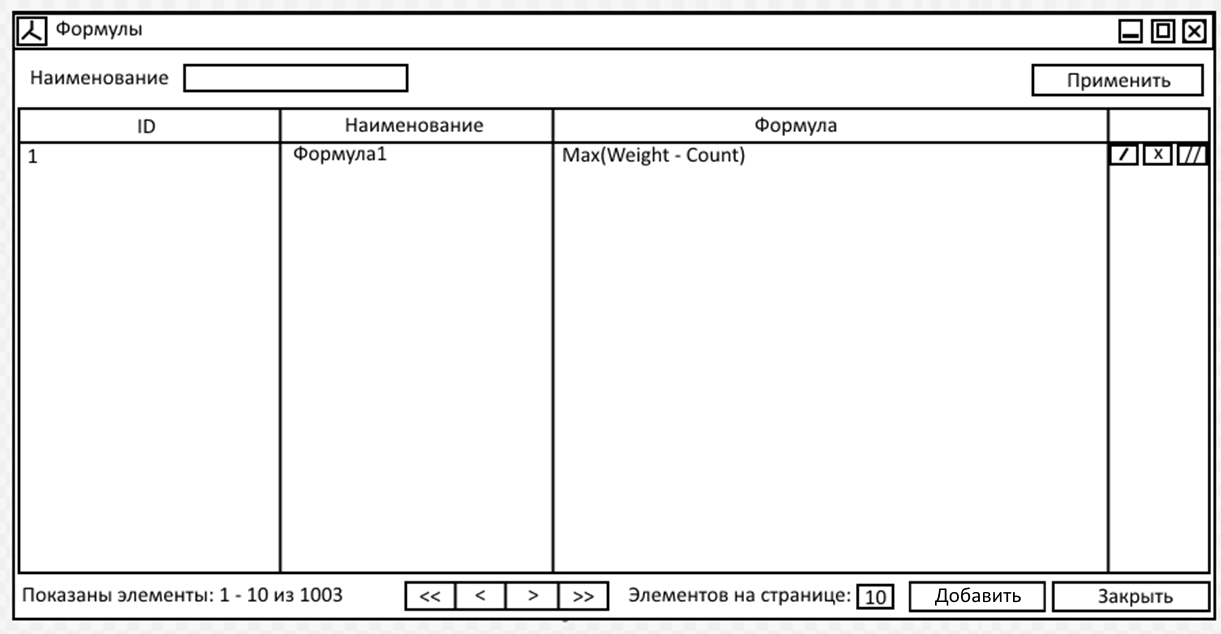
- Копировать

- Кнопка «Применить» на панели фильтров

- Элементы управления пагинатора

- Кнопка «Закрыть»

* + 1. Дизайн



* + 1. Поля
* Таблица: Список формул
* Фильтры:
  + Текстовое поле: Наименование
* Пагинатор:
  + Числовое поле: Кол-во элементов на странице
    1. Логика

Окно представляет собой список формул с блоком фильтров и пагинатором. Напротив каждой записи три кнопки – «Редактировать», «Копировать», «Удалить».

Общий элемент управления – кнопка Добавить.

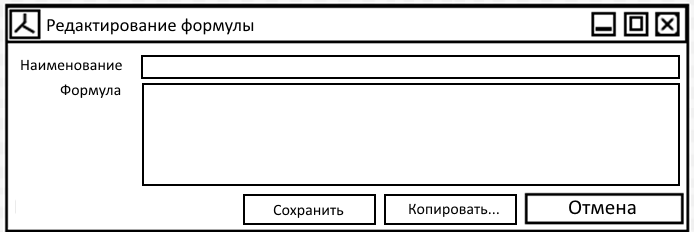
По нажатию на «Добавить» открывается страница добавления формулы.

По нажатию на «Редактировать» открывается страница редактирования выбранной формулы.

По нажатию на «Удалить» открывается окно подтверждения удаления формулы.

По нажатию на «Копировать» открывается окно добавления формулы с заполненными полями Наименование и Формула, скопированными из выбранной записи.

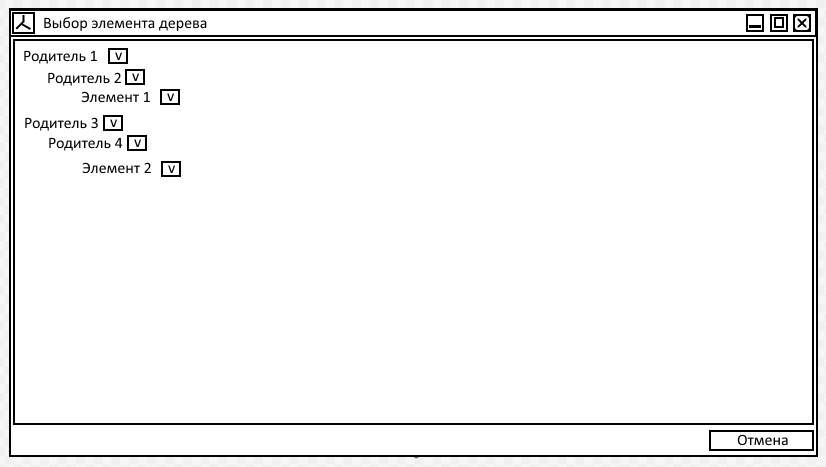
* 1. Окно редактирования формулы
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Отменить»
* Кнопка «Копировать»
  + 1. Дизайн



* + 1. Поля
* Наименование
* Формула
* Флаг «Установить по умолчанию»
  + 1. Логика

Окно предназначено для редактирования формулы, по кнопке «Сохранить» формула сохраняется в базу данных, по кнопке «Отмена» форма закрывается без сохранения, по кнопке «Копировать» форма закрывается и открывается новая форма со скопированными из текущей формы данными.

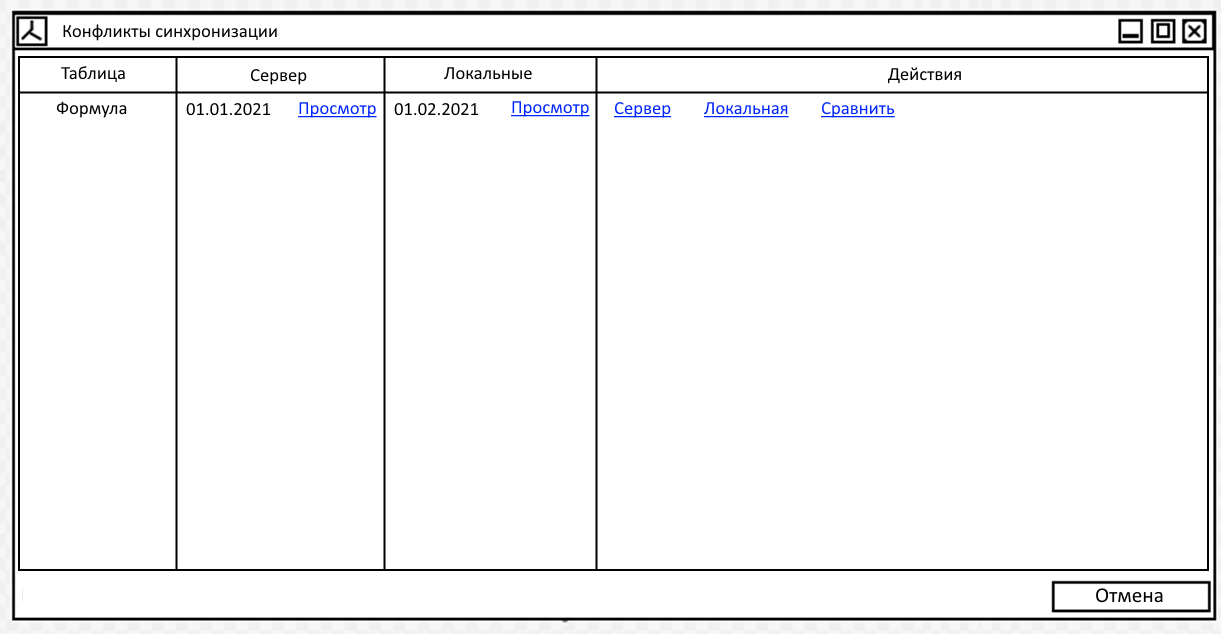
* 1. Окно выбора элемента дерева
     1. Управляющие элементы
* Кнопка «Выбрать» напротив каждого из элементов
* Кнопка «Отмена»
  + 1. Дизайн



* + 1. Поля
* Tree-компонент: список элементов дерева
  + 1. Логика

Окно содержит список элементов с кнопками выбора и кнопку «Отменить». По нажатию на кнопку «Выбрать» окно закрывается, в вызывающее окно возвращается id выбранного элемента. По нажатию на «Отменить» окно возвращается, не возвращая ничего.

* 1. Окно синхронизации
     1. Управляющие элементы
* Кнопка «Отмена»
* Кнопка «Просмотр» в колонке «Версия на сервере»
* Кнопка «Просмотр» в колонке «Локальная версия»
* Кнопки действий: «Принять серверный вариант», «Принять локальный вариант», «Сравнить версии и выбрать поля для сохранения»
  + 1. Дизайн



* + 1. Поля

- Таблица конфликтов синхронизации

* + 1. Логика

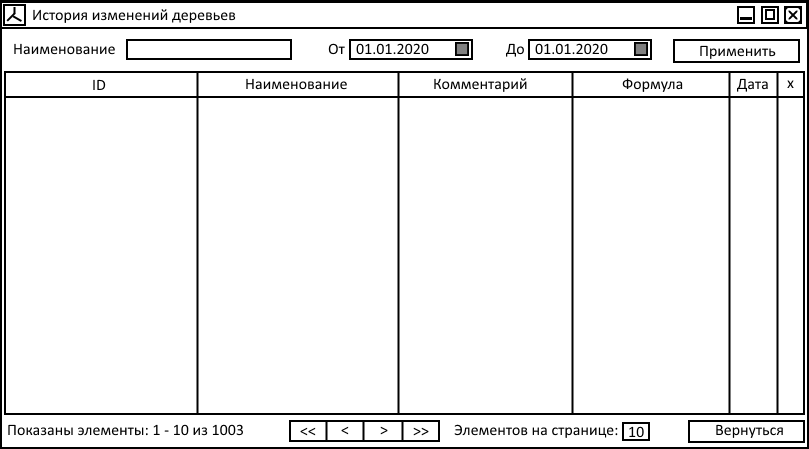
Окно открывается только в случае наличия конфликтов синхронизации с сервером. Конфликт возникает, если на сервере есть записи младше либо равные сохраняемой по номеру версии.

По кнопке «Просмотр» открывается окно просмотра соответствующей записи.

По кнопке выбора версии («Сервер», «Локальная») запись синхронизируется, по кнопке «Сравнить» открывается окно слияния.

По кнопке «Отмена» окно закрывается, на главном окне отображается напоминание о наличии в системе конфликтов синхронизации.

* 1. Окно истории изменений деревьев
     1. Управляющие элементы
* Кнопка «Вернуться»
* Блок фильтров: кнопка «Применить»
* Управляющие элементы пагинатора
  + 1. Дизайн



* + 1. Поля
* Таблица истории изменений
* Поля фильтров: «Наименование», поля дат изменений «От» и «До»
* Поле пагинатора «Кол-во элементов на странице»
  + 1. Логика

По открытию страницы отображается история изменений элементов по выбранному дереву за весь период по убыванию даты, в фильтрах возможно изменить эти параметры. По кнопке «Вернуться» форма закрывается и открывается предыдущая страница.

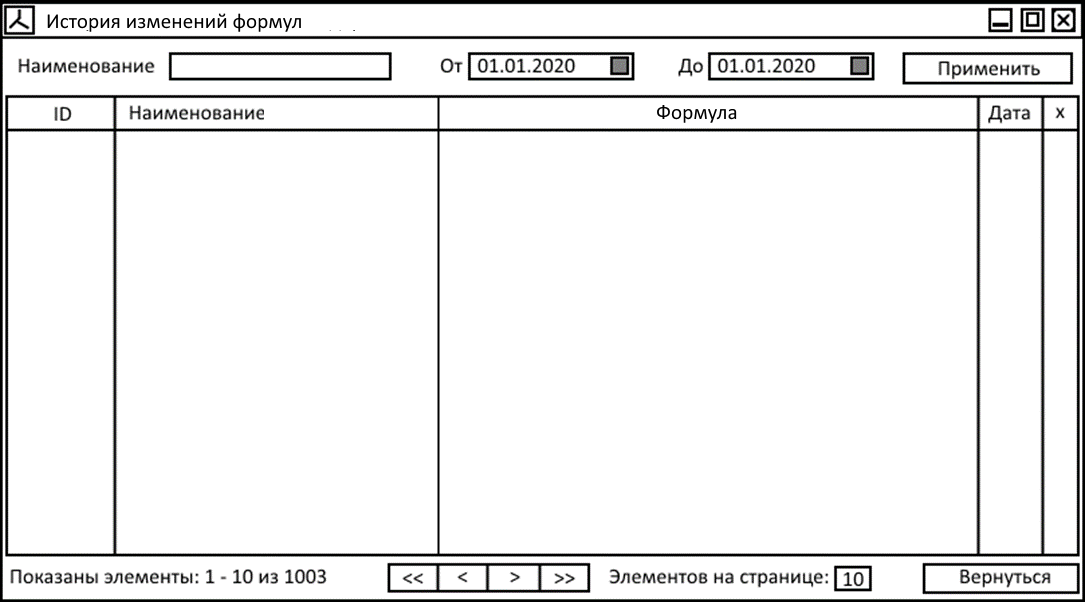
* 1. Окно истории изменений элементов деревьев
     1. Управляющие элементы
* Панель фильтров: кнопка «Показать»
* Кнопка «Вернуться»
* Управляющие элементы пагинатора
  + 1. Дизайн



* + 1. Поля
* Таблица истории изменений
* Поля фильтров:
  + Наименование: текстовое поле
  + От: поле даты
  + До: поле даты
* Поля пагинатора:
  + Кол-во элементов на странице
    1. Логика

По открытию страницы отображается история изменений элементов по выбранному дереву за весь период по убыванию даты, в фильтрах возможно изменить эти параметры. По кнопке «Вернуться» форма закрывается и открывается предыдущая страница.

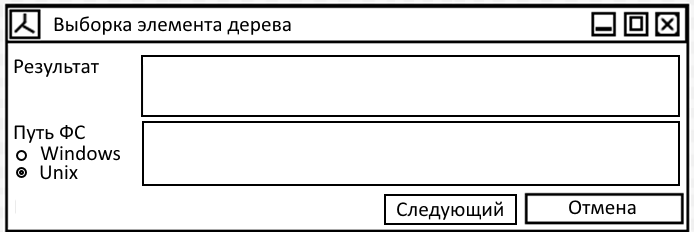
* 1. Окно истории изменений формул
     1. Управляющие элементы
* Кнопка «Показать» панели фильтров
* Кнопка «Вернуться»
* Управляющие элементы пагинатора
  + 1. Дизайн



* + 1. Поля
* Таблица истории изменений
* Поля фильтров:
  + Наименование: текстовое поле
  + От: поле даты
  + До: поле даты
* Поле пагинатора: кол-во элементов на странице
  + 1. Логика

По открытию страницы отображается история изменений формул за весь период по убыванию даты, в фильтрах возможно изменить эти параметры. По кнопке «Вернуться» форма закрывается и открывается предыдущая страница.

* 1. Окно выборки следующего элемента дерева
     1. Управляющие элементы
* Кнопка «Следующий»
* Кнопка «Закрыть»
  + 1. Дизайн



* + 1. Поля
* Поле результата (readonly)
* Путь в файловой системе
  + 1. Логика

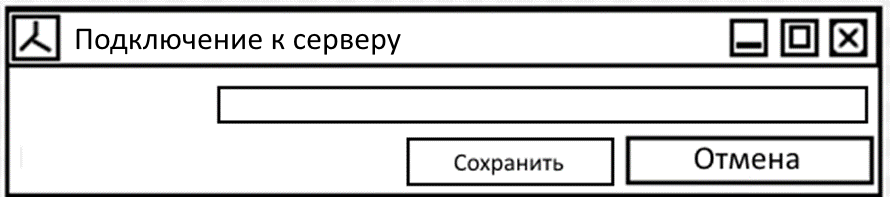
При открытии окна делается выборка следующего элемента дерева и увеличивается счетчик кол-ва выборов для выбранного элемента. Выбранный элемент отображается с полным путем от вершины дерева.

По кнопке «Следующий» делается выборка следующего элемента, по кнопке «Вернуться» окно закрывается.

* 1. Окно подключения к серверу
     1. Поля

- Текстовое поле: сервер (URL)

* + 1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Отмена»
  + 1. Дизайн



* + 1. Логика

По кнопке сохранить выполняется попытка подключения к серверу. В случае успеха строка подключения сохраняется в базе данных, окно закрывается. ПО кнопке Отмена окно закрывается без сохранения.

1. Web клиент
   1. Меню
      1. Главная

Открывает стартовую страницу

* + 1. Деревья

Открывает страницу «Деревья»

* + 1. Формулы

Открывает страницу «Формулы»

* 1. Пагинатор

Общий элемент для всех листовых страниц. Состоит из 4 управляющих элементов: кнопок «В начало», «На предыдущую страницу», «На следующую страницу» и «В конец» и поля ввода, где можно указать кол-во элементов, отображаемых на странице.

* 1. Главная страница (список деревьев)
     1. Управляющие элементы

- Кнопка “Добавить” - добавить дерево. Открывается страница добавления описания дерева

- Кнопка «Формулы» - открывается окно редактирования списка формул

- Напротив каждого элемента таблицы управляющие элементы (кнопки или ссылки):

- «Редактировать»

- «Удалить»

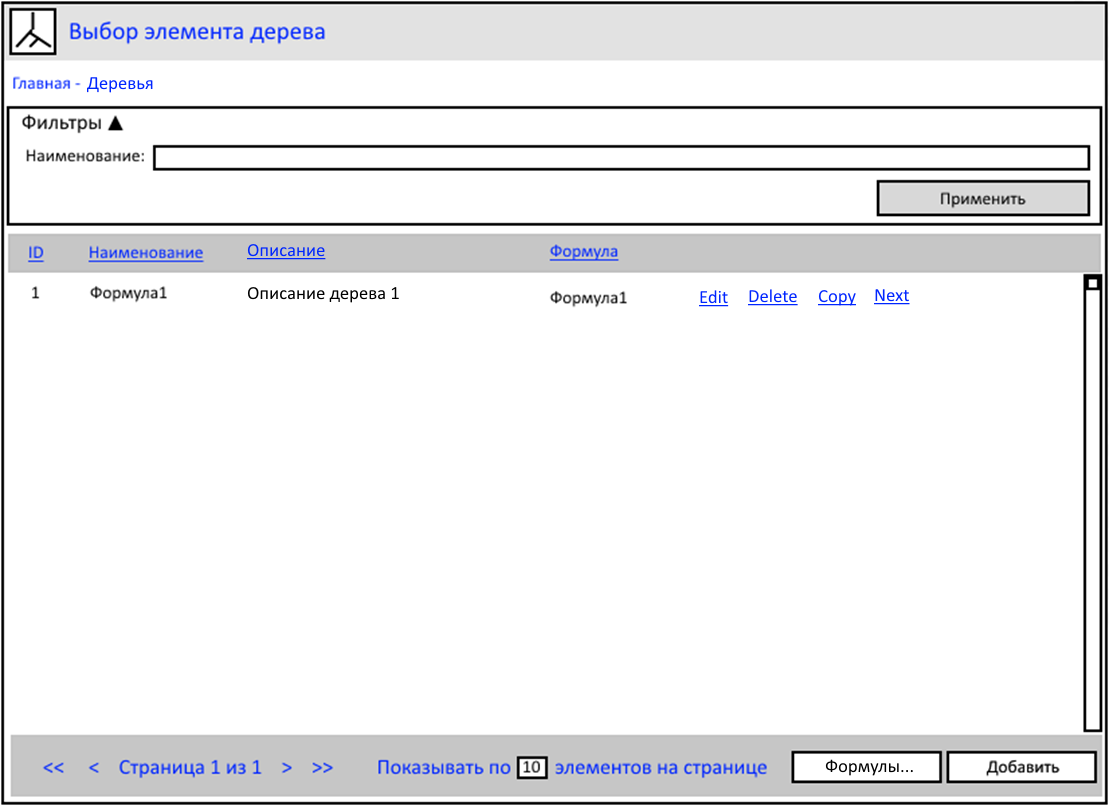
- «Копировать»

- «Получить следующий элемент»

- Управляющие элементы пагинатора

- Кнопка «Применить» блока фильтров

* + 1. Дизайн:



* + 1. Поля

Табличный список деревьев с постраничной отдачей. Колонки таблицы:

* + - * + Таблица: список деревьев
        + Поля фильтра:

Наименование

* + - * + Поля пагинатора
    1. Логика

Страница представляет собой список деревьев с постраничной отдачей, блоком фильтров (по полю «Наименование» и пагинатором).

По кнопке «Добавить» открывается окно добавления нового дерева.

По кнопке «Формулы» открывается окно списка формул.

Для каждого элемента списка:

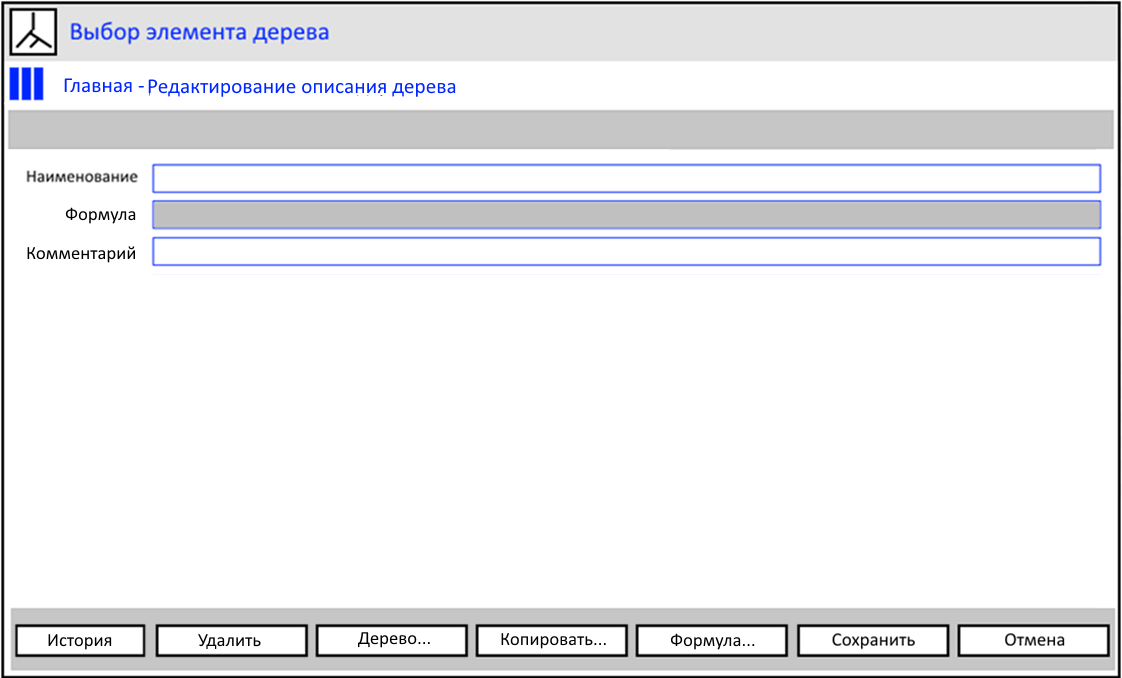
По кнопке «Редактировать» открывается окно редактирования описания дерева.

По кнопке «Удалить» открывается окно подтверждения удаления дерева.

По кнопке «Копировать» открывается окно добавления нового дерева – копии выбранного (в том числе элементов).

По кнопке «Следующий» открывается окно получения следующего элемента дерева, вычисленного по формуле (если элементов нет, возвращается ошибка)

* 1. Страница редактирования описания дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Копировать в новое дерево»
* Кнопка «Изменить формулу»
* Кнопка «Просмотреть историю изменений»
* Кнопка «Просмотр/Редактирование дерева»
* Кнопка «Удалить»
* Кнопка «Отменить»
  + 1. Поля
* Наименование
* Комментарий
* Формула (readonly)
  + 1. Дизайн



* + 1. Логика

Страница содержит два редактируемых поля, поле формула задается выбором нажатием на кнопку «Формула».

По кнопке «История» открывается окно истории изменений дерева.

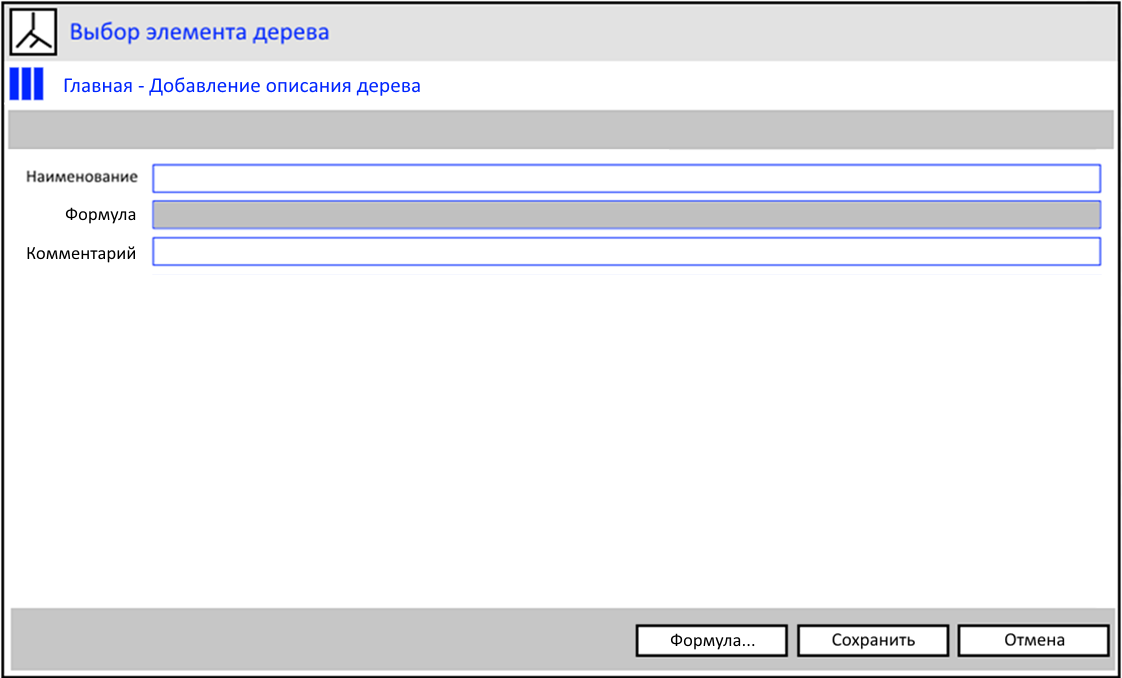
По кнопке «Удалить» открывается окно подтверждения удаления дерева.

По кнопке «Дерево» открывается окно редактирования дерева.

По кнопке «Копировать» открывается окно добавления описания дерева (в случае сохранения дерево будет скопировано в новое).

По кнопке «Сохранить» описание дерева сохраняется в базу. По кнопке «Отмена» форма закрывается без сохранения.

* 1. Страница добавления описания дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Выбрать формулу»
* Кнопка «Отменить»
  + 1. Поля
* Наименование
* Комментарий
* Формула (readonly)
  + 1. Дизайн

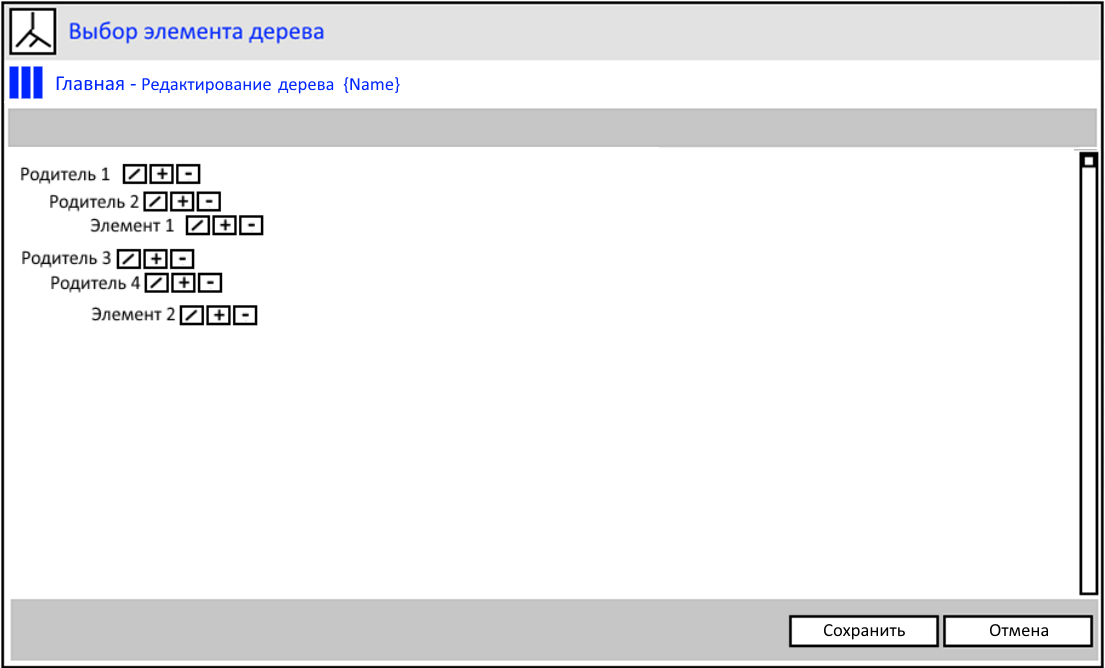


* + 1. Логика

Страница содержит два редактируемых поля, поле формула задается выбором нажатием на кнопку «Формула». Если формула не задана, сохранение не доступно.

По кнопке сохранить описание дерева сохраняется в базу. По кнопке отмена форма закрывается без сохранения.

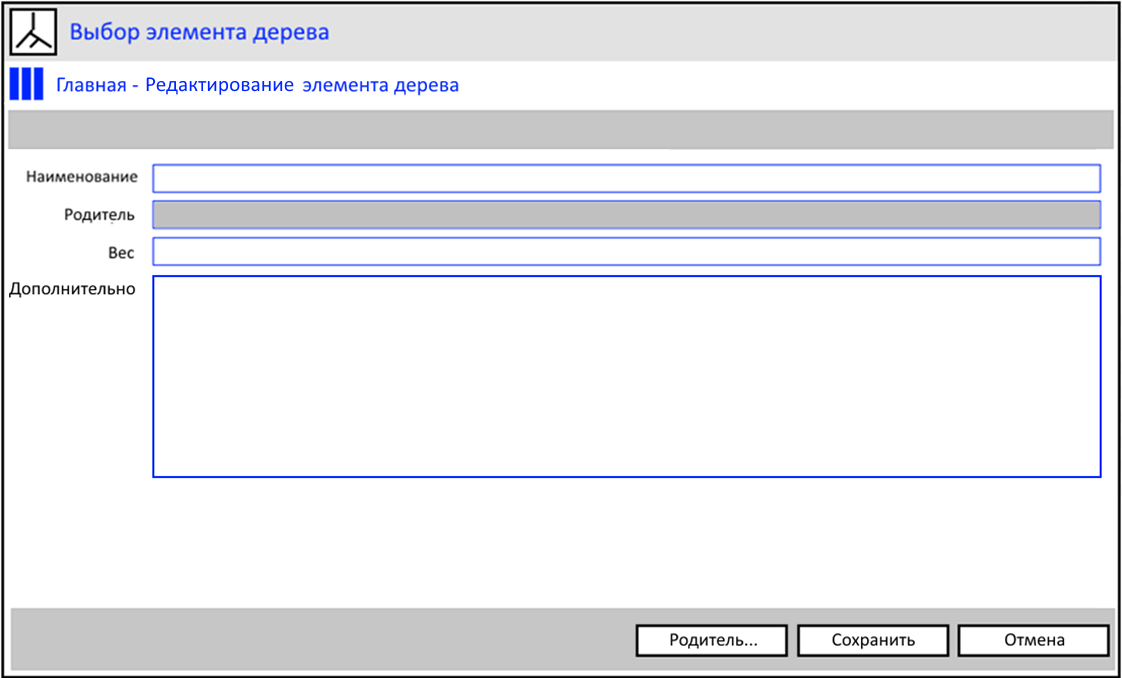
* 1. Страница редактирования дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Отменить»
* Напротив каждого элемента кнопки:
  + Редактировать
  + Добавить
  + Удалить
    1. Дизайн



* + 1. Поля
* Tree-элемент
  + 1. Логика

Страница представляет собой дерево, напротив каждого элемента имеются управляющие элементы (кнопки) «Редактировать», «Добавить» (дочерний элемент), «Удалить». По нажатию на каждый из элементов открывается соответствующее окно.

* 1. Страница редактирования элемента дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Родитель…»
* Кнопка «Отменить»
  + 1. Дизайн



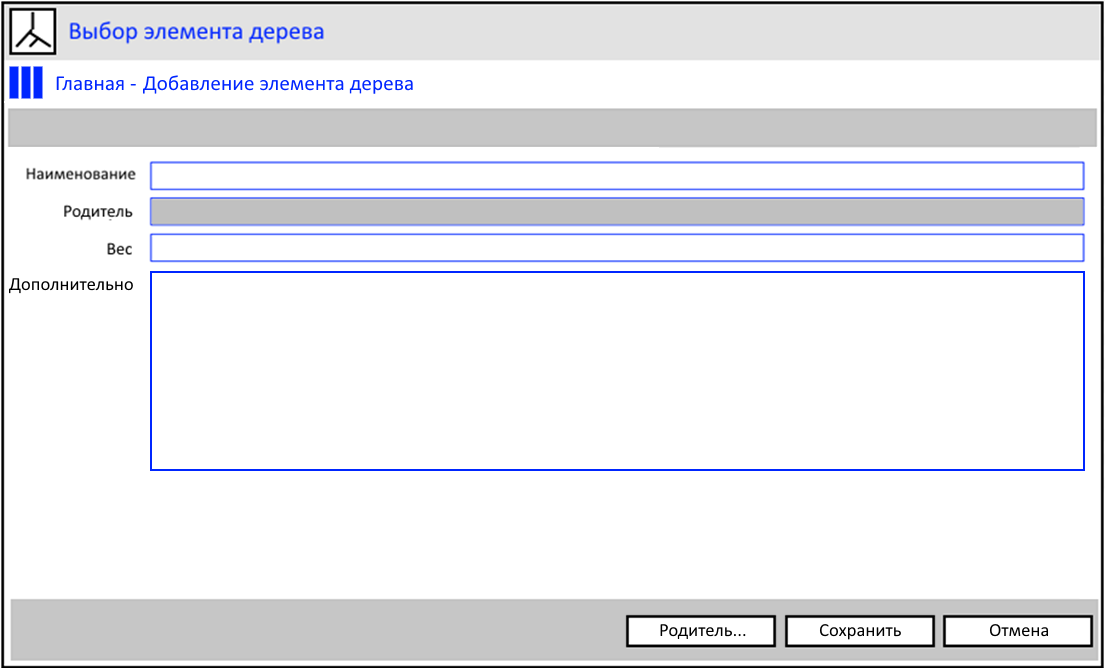
* + 1. Поля
* Наименование
* Родитель (readonly)
* Кол-во выборов (readonly)
* Вес
* Дополнительно
  + 1. Логика

На форме можно задать наименование, вес элемента и дополнительные поля (в формате json). Изменить родительский элемент можно по кнопке «Изменить родителя» (открывается окно выбора родительского элемента). По кнопке «Сохранить» элемент сохраняется на форме дерева, форма добавления закрывается. По кнопке «Отменить» элемент не добавляется, форма добавления закрывается.

Вес элемента – целое положительное число, минимум 1.

Дополнительно – поле в формате json.

* 1. Страница добавления элемента дерева
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Родитель…»
* Кнопка «Отменить»
  + 1. Меню – общий элемент
    2. Дизайн



* + 1. Поля
* Наименование
* Родитель (readonly)
* Вес
* Дополнительно
  + 1. Логика

На форме можно задать наименование, вес элемента и дополнительные поля (в формате json). Изменить родительский элемент можно по кнопке «Изменить родителя» (открывается окно выбора родительского элемента). По кнопке «Сохранить» элемент сохраняется на форме дерева, форма добавления закрывается. По кнопке «Отменить» элемент не добавляется, форма добавления закрывается.

Вес элемента – целое положительное число, минимум 1.

Дополнительно – поле в формате json.

* 1. Пагинатор

Представляет собой общий элемент для всех страниц со списками.

По изменению страницы или размера делается запрос обновления данных.

Информационные элементы:

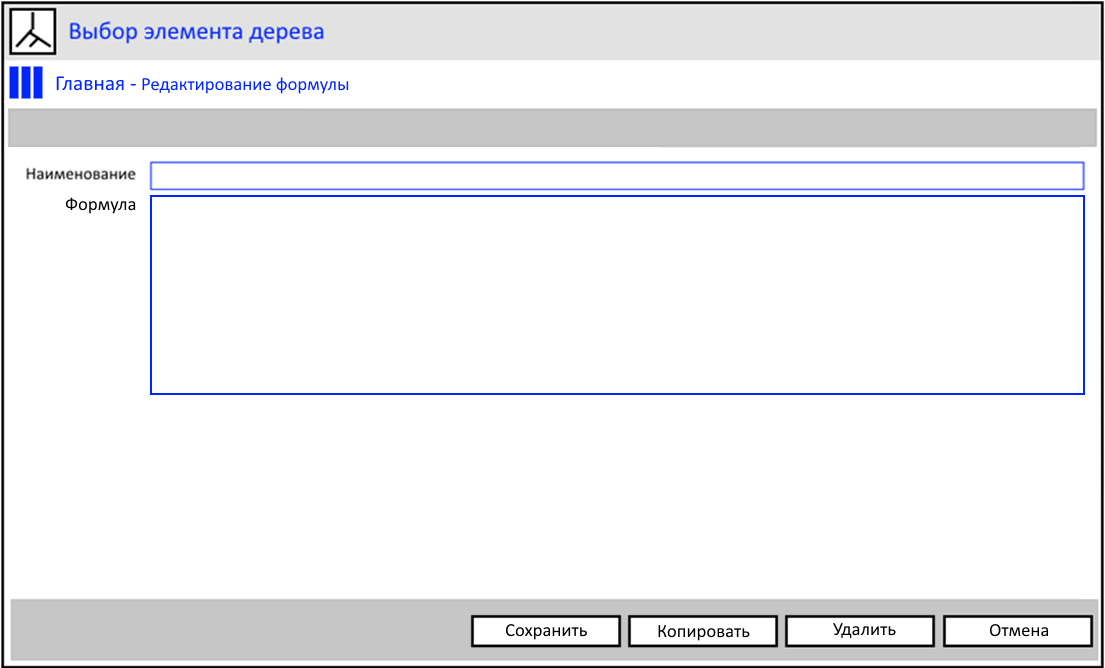
* Номер страницы (либо порядковые номера элементов от - до)
* Общее количество страниц

Управляющие элементы

* В начало (кнопка)
* В конец (кнопка)
* Предыдущая страница (кнопка)
* Следующая страница (кнопка)
* Количество на странице (выпадающий список)

Элементы расположены в одну строку

* 1. Общий дизайн (см. раздел «Дизайн» проекта)
  2. Страница редактирования формулы
     1. Управляющие элементы
* Кнопка «Сохранить»
* Кнопка «Копировать в новую формулу»
* Кнопка «Отменить»
* Кнопка «Удалить»
  + 1. Дизайн



* + 1. Поля
* Текстовое поле: Наименование
* Текстовое поле: Формула
* Логическое значение (флаг): Сделать по умолчанию
  + 1. Логика

Поле «Формула» - textarea (многострочное текстовое поле).

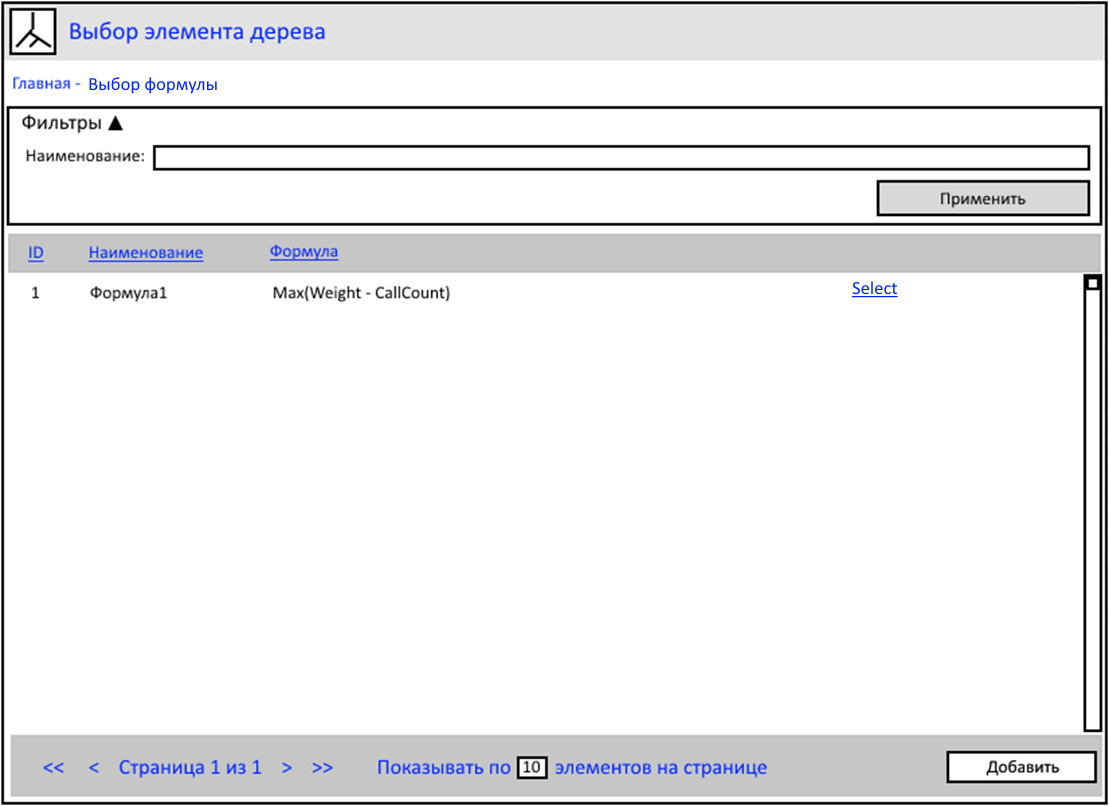
По нажатию на кнопку «Сохранить» формула сохраняется в базу, страница закрывается.

По нажатию на кнопку «Копировать» страница закрывается, открывается форма добавления формулы с копиями полей (Наименование генерируется новое как «наименование» + « копия»).

По нажатию на кнопку «Отмена» страница закрывается без сохранения, если были изменения – открывается форма подтверждения отмены.

По нажатию на кнопку «Удалить» формула удаляется из базы, страница закрывается, открывается форма подтверждения удаления. Если формула привязана к дереву, генерируется ошибка.

* 1. Страница выбора формулы
     1. Дизайн



* + 1. Поля
* Таблица: список формул
* Фильтр: Текстовое поле: Наименование
* Пагинатор: кол-во элементов на странице
  + 1. Логика

Форма предназначена для выбора формулы, вызывается с формы редактирования описания дерева. По нажатия кнопки выбора напротив выбранной формулы возвращает в вызывающую форму Id выбранной формулы.

* + 1. Управляющие элементы

- Кнопки выбора формулы напротив каждой формулы

- Кнопка «Применить» блока фильтров

- Управляющие элементы пагинатора

* 1. Страница редактирования списка формул
     1. Управляющие элементы

- Кнопка «Добавить»

- Напротив каждой формулы кнопки:

- Редактировать

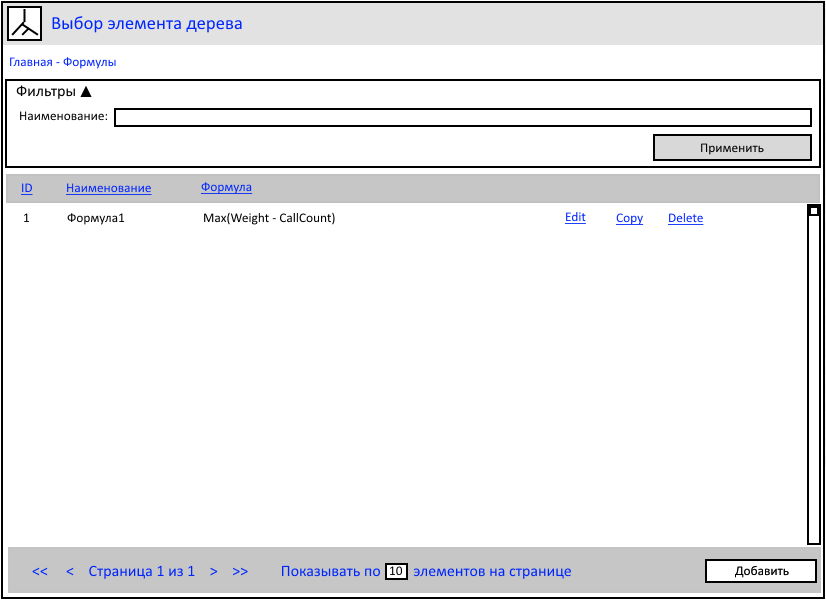
- Удалить

- Копировать

- Кнопка «Применить» в блоке фильтров

Элементы пагинатора (общий элемент)

* + 1. Поля
* Таблица: список формул
* Фильтры:
  + Текстовое поле: Наименование
* Текстовое поле: Кол-во записей на странице
  + 1. Дизайн



* + 1. Логика

Страница представляет собой список формул с блоком фильтров и пагинатором. Напротив каждой записи три кнопки – «Редактировать», «Копировать», «Удалить».

Общий элемент управления – кнопка Добавить.

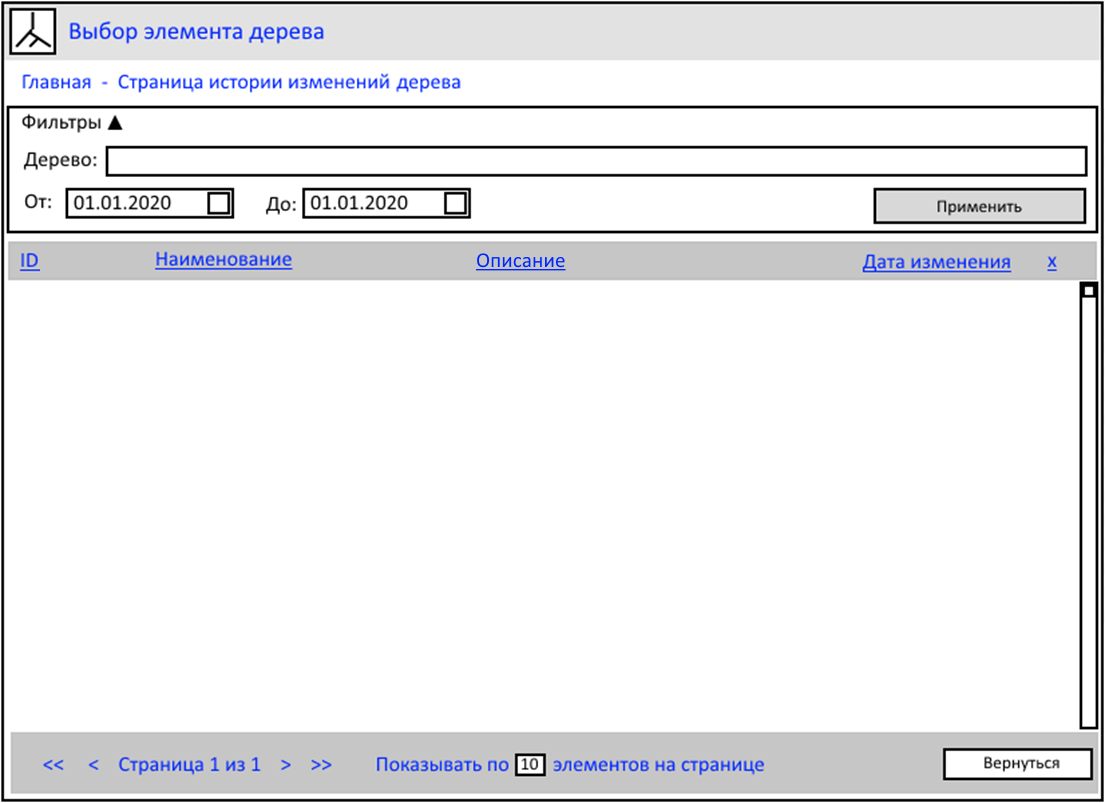
По нажатию на «Добавить» открывается страница добавления формулы.

По нажатию на «Редактировать» открывается страница редактирования выбранной формулы.

По нажатию на «Удалить» открывается окно подтверждения удаления формулы.

По нажатию на «Копировать» открывается окно добавления формулы с заполненными полями «Наименование» и «Формула», скопированными из выбранной записи.

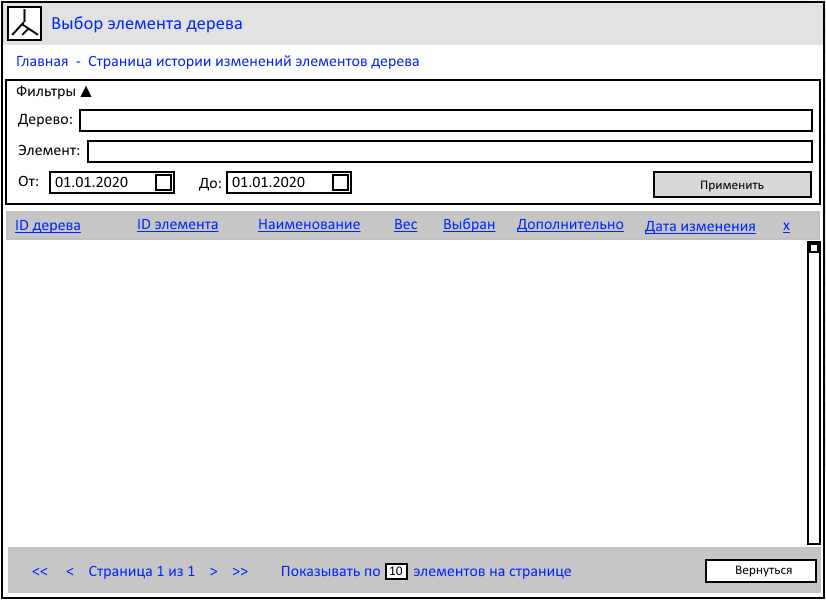
* 1. Страница истории изменений деревьев
     1. Управляющие элементы
* Кнопка «Применить фильтр»
* Кнопка «Вернуться»
* Управляющие элементы пагинатора
  + 1. Поля
       - Таблица: Список истории изменений
       - Поля фильтров:
         * Текстовое поле: Дерево
         * Поле даты: От
         * Поле даты: До
       - Числовое поле: Количества отображаемых элементов
    2. Дизайн



* + 1. Логика

По открытию страницы отображается история по выбранному дереву за весь период по убыванию даты, в фильтрах возможно изменить эти параметры. По кнопке «Вернуться» форма закрывается и открывается предыдущая страница.

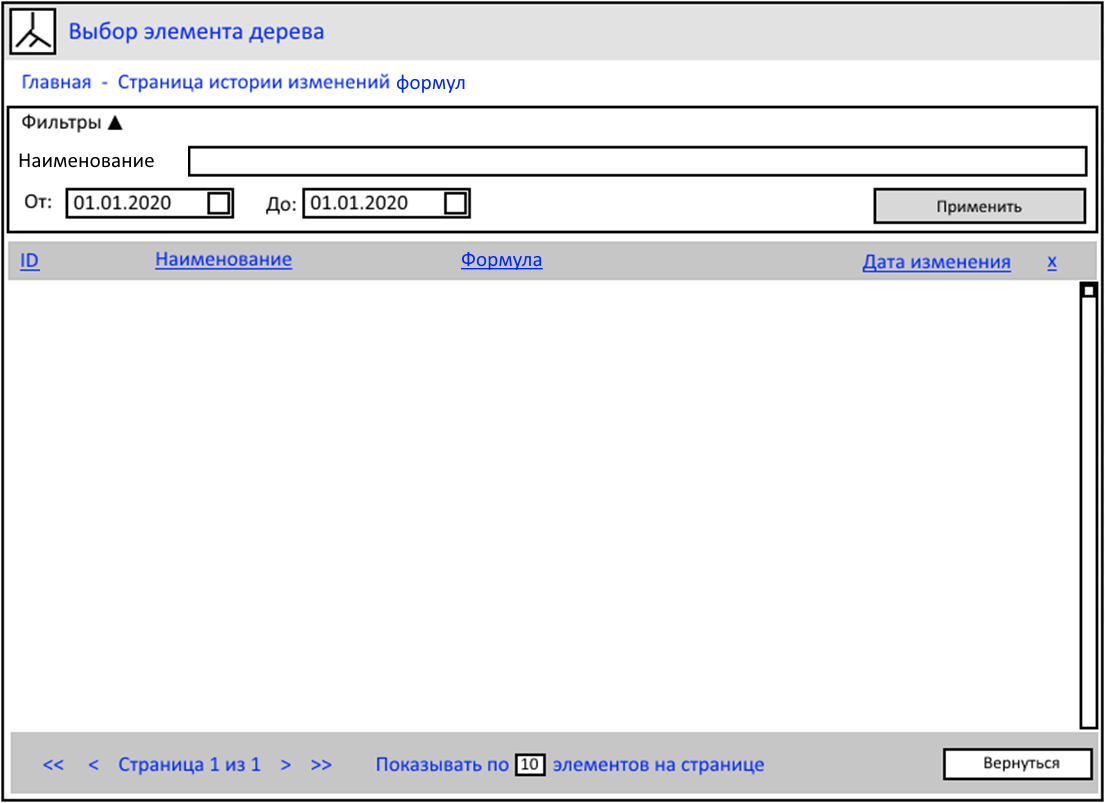
* 1. Страница истории изменений элементов деревьев
     1. Управляющие элементы
* Фильтры: Кнопка «Применить фильтр»
* Кнопка «Вернуться»
* Управляющие элементы пагинатора
  + 1. Поля
       - Таблица: Список истории изменений
       - Поля фильтров:
         * Текстовое поле: Дерево
         * Текстовое поле: Элемент
         * Поле даты: От
         * Поле даты: До
       - Пагинатор:
         * Числовое поле: кол-во элементов на странице
    2. Дизайн



* + 1. Логика

По открытию страницы отображается история изменений элементов по выбранному дереву за весь период по убыванию даты, в фильтрах возможно изменить эти параметры. По кнопке «Вернуться» форма закрывается и открывается предыдущая страница.

* 1. Страница истории изменений формул
     1. Управляющие элементы
* Кнопка «Вернуться»
* Блок фильтров: Кнопка «Применить»
* Управляющие элементы пагинатора
  + 1. Поля
       - Таблица: Список истории изменений
       - Фильтры
         * Текстовое поле: Наименование
         * Поле даты: От
         * Поле даты: До
       - Пагинатор
         * Числовое поле: кол-во элементов на странице
    2. Дизайн



* + 1. Логика

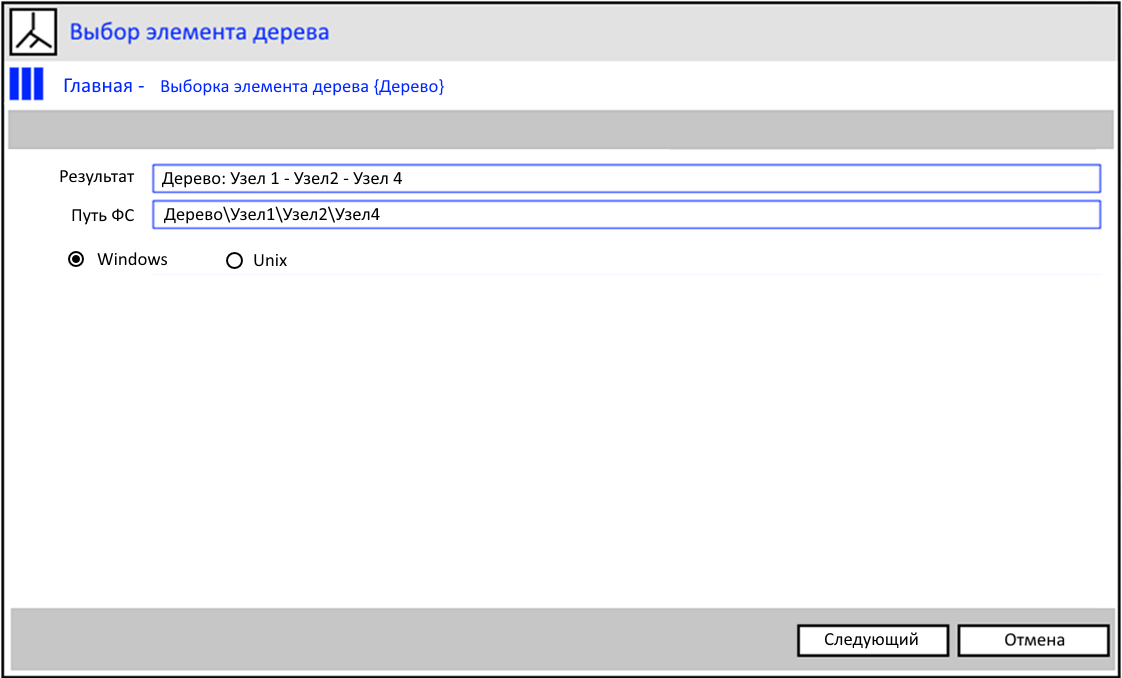
По открытию страницы отображается история изменений формул за весь период по убыванию даты, в фильтрах возможно изменить эти параметры. По кнопке «Вернуться» форма закрывается и открывается предыдущая страница.

* 1. Страница выборки следующего элемента дерева (БД)
     1. Управляющие элементы

- Кнопка «Вернуться»

- Кнопка «Следующий»

* + 1. Поля
       - Текстовое поле результата выборки элемента (полный путь от вершины дерева)
       - Текстовое поле результата выборки элемента в виде пути ФС
    2. Дизайн



* + 1. Логика

При открытии окна делается выборка следующего элемента дерева и увеличивается счетчик кол-ва выборов для выбранного элемента. Выбранный элемент отображается с полным путем от вершины дерева, в отдельном поле – отображение в формате файловой системы.

По кнопке «Следующий» делается выборка следующего элемента, по кнопке «Вернуться» окно закрывается.

1. Установщик
   1. Сервер Windows

Для построения установщика используется WixInstaller. Содержит следующие окна:

* + 1. Описание
    2. Лицензионное соглашение
    3. Выбор папки установки
    4. Выбор поставщика базы данных
    5. Подключение к базе данных
    6. Выбор режима работы (служба/приложение)
    7. Установка
    8. Выход
  1. Сервер Linux

Приложение поставляется в виде архива tar.gz. Конфигурируется вручную, описание настройки и лицензионное соглашение располагаются в файле readme.

* 1. Десктоп Windows

Для построения установщика используется WixInstaller. Содержит следующие окна:

* + 1. Описание
    2. Лицензионное соглашение
    3. Выбор папки установки
    4. Установка пароля базы данных (опционально)
    5. Установка параметров подключения к серверу (опционально)
    6. Установка
    7. Выход
  1. Сервер Docker

Приложение поставляется в виде контенйнера. Конфигурируется вручную, описание настройки и лицензионное соглашение располагаются в файле readme.

1. Тестирование
   1. Модульное тестирование
      1. Сервер
         * Тесты репозиториев
           + Метод Add

Проверяется работа метода Add репозитория для любой из моделей. Последовательность теста:

* Создание окружения
* Добавление сущности
* Получение сущности по Id
* Проверка на not null
* Проверка на идентичность полей базовой и сохраненной моделей
  + - * + Метод Get

Проверяется работа метода Get репозитория для любой из моделей. Последовательность теста:

* Создание окружения
* Добавление нескольких сущностей
* Получение сущностей по фильтру
* Проверка на соответствие полученных сущностей фильтру
  + - * + Метод GetItem

Проверяется работа метода GetItem репозитория для любой из моделей. Последовательность теста:

* Создание окружения
* Добавление сущности
* Получение сущности по Id
* Проверка на Not null
  + - * + Метод Update

Проверяется работа метода Update репозитория для любой из моделей. Последовательность теста:

* Создание окружения
* Добавление сущности
* Получение сущности по Id
* Проверка на not null
* Проверка на идентичность полей базовой и сохраненной моделей
* Изменение сущности
* Получение сущности по Id
* Проверка на not null
* Проверка на идентичность ожидаемых (измененных) полей и полей сохраненной модели
  + - * + Метод Delete

Проверяется работа метода Delete репозитория для любой из моделей. Последовательность теста:

* Создание окружения
* Добавление сущности
* Получение сущности по Id
* Проверка на not null
* Проверка на идентичность полей базовой и сохраненной моделей
* Удаление сущности
* Получение сущности по Id
* Проверка на null
  + - * Тесты API
        + Дерево

Метод получения списка деревьев

Проверяется работа метода получения списка деревьев согласно переданным фильтрам. Последовательность теста:

* Создание окружения
* Добавление нескольких деревьев
* Получение деревьев по фильтру через API
* Проверка на соответствие полученных деревьев фильтру

Метод получения дерева по Id

Проверяется работа метода получения дерева по идентификатору. Последовательность теста:

* Создание окружения
* Добавление дерева
* Получение дерева по Id через API
* Проверка на not null

Метод получения всех элементов дерева

Проверяется работа метода получения всех элементов дерева по идентификатору дерева. Последовательность теста:

* Создание окружения
* Добавление дерева
* Добавление нескольких элементов дерева
* Получение дерева по Id через API
* Проверка на not null
* Получение всех элементов дерева по Id через API
* Проверка на соответствие элементов (по количеству)

Метод добавления дерева

Проверяется работа метода добавления дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод API
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому

Метод изменения дерева

Проверяется работа метода изменения дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод API
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
* Изменение дерева (в т.ч. элементов)
* Вызов метода изменения дерева через API
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов измененному

Метод изменения элементов дерева

Проверяется работа метода изменения элементов дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод API
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
* Изменение элементов дерева (перенос в другую ветвь, переименование, добавление нового, удаление)
* Вызов метода изменения элемента дерева через API
* Получение элементов дерева через репозиторий
* Проверка что элементы были изменены

Метод удаления дерева

Проверяется работа метода удаления дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод API
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
* Вызов метода удаления дерева через API
* Получение дерева через репозиторий
* Проверка IsDeleted = true

Методы получения списка изменений

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через репозиторий
* Получение дерева, его элементов и списка изменений дерева и элементов через API
* Проверка на соответствие дерева и его элементов базовому, а также соответствие изменений произведенным изменениям
* Изменение элементов дерева (перенос в другую ветвь, переименование, добавление нового, удаление)
* Получение дерева, его элементов и списка изменений дерева и элементов через API
* Проверка на соответствие дерева и его элементов базовому, а также соответствие изменений произведенным изменениям
* Вызов метода удаления дерева через API
* Получение дерева, его элементов и списка изменений дерева и элементов через API
* Проверка на соответствие дерева и его элементов базовому, а также соответствие изменений произведенным изменениям
  + - * + Формула

Метод получения списка формул

Проверяется работа метода получения списка формул согласно переданным фильтрам. Последовательность теста:

* Создание окружения
* Добавление нескольких формул
* Получение формул по фильтру через API
* Проверка на соответствие полученных формул фильтру

Метод получения формулы по id

Проверяется работа метода получения формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через репозиторий
* Получение формулы через API
* Проверка на not null

Метод добавления формулы

Проверяется работа метода добавления формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через метод API
* Получение формулы через репозиторий
* Проверка на соответствие формулы базовой

Метод изменения формулы

Проверяется работа метода изменения формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через метод API
* Получение формулы через репозиторий
* Проверка на соответствие формулы базовой
* Изменение формулы
* Вызов метода изменения формулы через API
* Получение формулы через репозиторий
* Проверка на соответствие формулы измененной

Метод удаления формулы

Проверяется работа метода удаления формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через метод API
* Получение формулы через репозиторий
* Проверка на соответствие формулы базовой
* Вызов метода удаления формулы через API
* Получение формулы через репозиторий
* Проверка на null

Методы получения списка изменений

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление формулы через репозиторий
* Получение формулы и списка изменений формулы через API
* Проверка на соответствие формулы базовому, а также соответствие изменений произведенным изменениям
* Изменение формулы
* Получение формулы и списка изменений формулы через API
* Проверка на соответствие формулы базовой, а также соответствие изменений произведенным изменениям
* Вызов метода удаления формулы через API
* Получение формулы и списка изменений формулы через API
* Проверка на null, а также соответствие изменений произведенным изменениям
  + - * + Метод выборки элемента (дерево в базе)

Проверяется работа метода расчета через API (дерево в базе). Последовательность теста:

* Создание окружения
* Создание дерева, формулы (с однозначным результатом выборки)
* Отправка на расчет нескольких запросов
* Проверка правильности расчета
  + - * + Метод выборки элемента (дерево в запросе)

Проверяется работа метода расчета через API (дерево в запросе). Последовательность теста:

* Создание окружения
* Отправка на расчет нескольких запросов с однозначным результатом выборки
* Проверка правильности расчета
  + - * + Библиотека расчета по формуле

Метод расчета

Проверяется работа метода расчета. Последовательность теста:

* Создание окружения
* Отправка на расчет нескольких формул с однозначным результатом
* Проверка правильности расчета
  + - * Сервис сервер
        + GetTrees

Проверяется работа метода получения списка деревьев согласно переданным фильтрам. Последовательность теста:

* Создание окружения
* Добавление нескольких деревьев
* Получение деревьев по фильтру через API
* Проверка на соответствие полученных деревьев фильтру
  + - * + GetFormulas

Проверяется работа метода получения списка формул согласно переданным фильтрам. Последовательность теста:

* Создание окружения
* Добавление нескольких формул
* Получение формул по фильтру через API
* Проверка на соответствие полученных формул фильтру
  + - * + GetTreeItems

Проверяется работа метода получения списка элементов дерева. Последовательность теста:

* Создание окружения
* Добавление дерева, нескольких элементов
* Получение элементов через API
* Проверка на соответствие полученных элементов добавленным
  + - * + AddTree

Проверяется работа метода добавления дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод AddTree
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
  + - * + GetTree

Проверяется работа метода получения дерева по Id. Последовательность теста:

* Создание окружения
* Добавление формулы, дерева
* Получение дерева по Id через метод
* Проверка на not null
  + - * + GetTreeItem

Проверяется работа метода получения элемента дерева по Id. Последовательность теста:

* Создание окружения
* Добавление формулы, дерева
* Добавление элемента дерева
* Получение элемента дерева по Id через метод
* Проверка на not null
  + - * + GetFormula

Проверяется работа метода получения формулы по Id. Последовательность теста:

* Создание окружения
* Добавление формулы
* Получение формулы по Id через метод
* Проверка на not null
  + - * + UpdateTreeItems

Проверяется работа метода изменения элементов дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод сервиса
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
* Изменение элементов дерева (перенос в другую ветвь, переименование, добавление нового, удаление)
* Вызов метода изменения элемента дерева через API
* Получение элементов дерева через репозиторий
* Проверка что элементы были изменены
  + - * + AddFormula

Проверяется работа метода добавления формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через метод
* Получение формулы через репозиторий
* Проверка на соответствие формулы базовой
  + - * + UpdateTree

Проверяется работа метода обновления дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через метод AddTree
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
* Изменение дерева
* Проверка на соответствие дерева измененному
  + - * + UpdateTreeItem

Проверяется работа метода обновления элемента дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через репозиторий
* Изменение одного элемента через метод
* Получение элемента через репозиторий
* Проверка на соответствие элемента измененному
  + - * + UpdateFormula

Проверяется работа метода изменения формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через репозиторий
* Обновление формулы через метод
* Получение формулы через репозиторий
* Проверка на соответствие формулы измененной
  + - * + UpdateTreeItem

Проверяется работа метода изменения элемента дерева. Последовательность теста:

* Создание окружения
* Добавление формулы, дерева, элемента через репозиторий
* Обновление элемента через метод
* Получение элемента через репозиторий
* Проверка на соответствие полученного элемента отправленному
  + - * + DeleteTree

Проверяется работа метода удаления дерева. Последовательность теста:

* Создание окружения
* Добавление дерева с несколькими элементами через репозиторий
* Получение дерева и его элементов через репозиторий
* Проверка на соответствие дерева и его элементов базовому
* Вызов метода удаления дерева через метод
* Получение дерева через репозиторий
* Проверка IsDeleted = true
  + - * + DeleteTreeItem

Проверяется работа метода удаления элемента дерева. Последовательность теста:

* Создание окружения
* Добавление формулы, дерева, элемента через репозиторий
* Удаление элемента через метод
* Получение элемента через репозиторий
* Проверка IsDeleted = true
  + - * + DeleteFormula

Проверяется работа метода удаления формулы. Последовательность теста:

* Создание окружения
* Добавление формулы через репозиторий
* Удаление формулы через метод
* Получение формулы через репозиторий
* Проверка IsDeleted = true
  + - * + Calculate

Проверяется работа метода расчета. Последовательность теста:

* Создание окружения
* Создание дерева, формулы (с однозначным результатом выборки)
* Отправка на расчет нескольких запросов
* Проверка правильности расчета
  + - * + CalculateCustom

Проверяется работа метода расчета. Последовательность теста:

* Создание окружения
* Отправка на расчет нескольких запросов с однозначным результатом выборки
* Проверка правильности расчета

GetTreeHistory

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление формулы и дерева через репозиторий
* Получение дерева и списка изменений формулы через метод
* Проверка на соответствие дерева базовому, а также соответствие изменений произведенным изменениям
* Изменение дерева
* Получение дерева и списка изменений формулы через метод
* Проверка на соответствие дерева базовому, а также соответствие изменений произведенным изменениям
* Вызов метода удаления дерева через репозиторий
* Получение дерева и списка изменений через метод
* Проверка на null, а также соответствие изменений произведенным изменениям

GetFormulaHistory

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление формулы через репозиторий
* Получение списка изменений формулы через метод
* Проверка на соответствие изменений произведенным изменениям
* Изменение формулы
* Получение формулы и списка изменений формулы через метод
* Проверка на соответствие формулы базовой, а также соответствие истории изменений произведенным изменениям
* Вызов метода удаления формулы через репозиторий
* Получение формулы и списка изменений через метод
* Проверка на null, а также соответствие изменений произведенным изменениям

GetTreeItemHistory

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление формулы, дерева и нескольких элементов через репозиторий
* Получение списка изменений элементов через метод
* Проверка на соответствие изменений произведенным изменениям
* Изменение нескольких элементов через репозиторий
* Получение списка изменений через метод
* Проверка на соответствие изменений произведенным изменениям
* Вызов метода удаления элемента через репозиторий
* Получение списка изменений через метод
* Проверка на соответствие изменений произведенным изменениям

GetTreeChanges

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление формулы и дерева через репозиторий
* Получение дерева и списка изменений формулы через метод
* Проверка на соответствие дерева базовому, а также соответствие изменений произведенным изменениям
* Изменение дерева
* Получение дерева и списка изменений формулы через метод
* Проверка на соответствие дерева базовому, а также соответствие изменений произведенным изменениям
* Вызов метода удаления дерева через репозиторий
* Получение дерева и списка изменений через метод
* Проверка на null, а также соответствие изменений произведенным изменениям
  + 1. Клиент
       - Сервис (методы проверяются в режиме ThickOnly – остальные режимы тестируются в интеграционных тестах)
         * GetTree

Проверяется работа метода получения дерева по Id. Последовательность теста:

* Создание окружения
* Добавление формулы, дерева через репозиторий
* Получение дерева по Id через метод
* Проверка на not null
  + - * + GetTrees

Проверяется работа метода получения списка деревьев согласно переданным фильтрам. Последовательность теста:

* Создание окружения
* Добавление нескольких деревьев
* Получение деревьев по фильтру через API
* Проверка на соответствие полученных деревьев фильтру

GetTreeHistory

Проверяется работа метода получения списка изменений. Последовательность теста:

* Создание окружения
* Добавление формулы и дерева через репозиторий
* Получение дерева и списка изменений формулы через метод
* Проверка на соответствие дерева базовому, а также соответствие изменений произведенным изменениям
* Изменение дерева
* Получение дерева и списка изменений формулы через метод
* Проверка на соответствие дерева базовому, а также соответствие изменений произведенным изменениям
* Вызов метода удаления дерева через репозиторий
* Получение дерева и списка изменений через метод
* Проверка на null, а также соответствие изменений произведенным изменениям
  + 1. Web-клиент
    2. Общие библиотеки
       - FormulaCalc. Проверяется работа метода Calculate.

Последовательность теста:

* + Создание окружения
  + Отправка на расчет нескольких запросов с однозначным результатом выборки
  + Проверка правильности расчета
    - * DeployBD. Проверяется работа библиотеки раскатки БД

Последовательность теста:

* Из тестовых скриптов раскатывается база данных, проверяется правильность раскатки и обновления.
* База данных удаляется
  1. Интеграционное тестирование
     1. Клиент
        + Сценарий 1
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы (однозначный результат, без рандома)
* Добавление дерева и элементов с привязкой к формуле
* Выборка нескольких значений, сравнение с контрольными
  + - * Сценарий 2
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы (с рандомом)
* Добавление дерева и элементов с привязкой к формуле
* Выборка нескольких значений (статистически значимое количество)
* Проверка равномерного распределения (с погрешностью)
  + - * Сценарий 3
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление нерабочей формулы
* Добавление дерева и элементов с привязкой к формуле
* Выборка значения
* Проверка правильного исключения
  + - * Сценарий 4
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы
* Добавление дерева и элементов с привязкой к формуле
* Попытка удаления формулы
* Проверка правильного исключения
  + - * Сценарий 5
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы
* Добавление дерева и элементов с привязкой к формуле
* Отключение сервера
* Изменение дерева и элементов на клиенте и сервере (через базу данных либо через сервер с другим адресом)
* Включение сервера
* Ожидание предложения синхронизации
* Синхронизация
* Проверка правильности данных на сервере
  + 1. Web-клиент.
       - Сценарий 1
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы (однозначный результат, без рандома)
* Добавление дерева и элементов с привязкой к формуле
* Выборка нескольких значений, сравнение с контрольными
  + - * Сценарий 2
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы (с рандомом)
* Добавление дерева и элементов с привязкой к формуле
* Выборка нескольких значений (статистически значимое количество)
* Проверка равномерного распределения (с погрешностью)
  + - * Сценарий 3
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление нерабочей формулы
* Добавление дерева и элементов с привязкой к формуле
* Выборка значения по формуле
* Проверка правильного исключения
  + - * Сценарий 4
* Создание окружения (деплой БД, запуск сервера, клиента)
* Добавление формулы
* Добавление дерева и элементов с привязкой к формуле
* Попытка удаления формулы
* Проверка правильного исключения
  1. Ручное тестирование

Тесты из интеграционного тестирования

* 1. Регрессионное тестирование

Тесты из интеграционного тестирования