

Реферат

программный комплекс

Конструктор концепций

(Версия 2.0)

Авторы:

Иванов Дмитрий Юрьевич

Набоков Сергей Анатольевич

2020 год

Введение

Мы живем в мире победивших лириков и биржевой экономики беспощадно попирающих физические основы мироздания, в мире, где систематизация данных в общем случае сводится к автоматизации операций связанных с движением средств и материально технических ресурсов от сырья через складские запасы к конечному продукту плюс бухгалтерский учет и налоговая отчетность, при этом огромный пласт задач по систематизации и учету технических, технологических, научных, социальных, природных процессов или процессов производства в собственном небольшом деле, остается далеко за гранью понимания вышеописанных систем. Это темная область, где безраздельно правит Excel и однопользовательские СУБД с низким порогом вхождения пользователей. Бесчисленное множество разрозненных данных, сведенных в электронные таблицы в попытках систематизировать, учесть, сохранить, сопоставить и повторить или не повторить, подготовить и передать из одной системы в другую, найти лучшее решение из множества доступных вариантов. Все это опыт необходимый в любом деле, который нужно сохранить и использовать, чтобы получить новый опыт, ибо только так можно достичь совершенства.

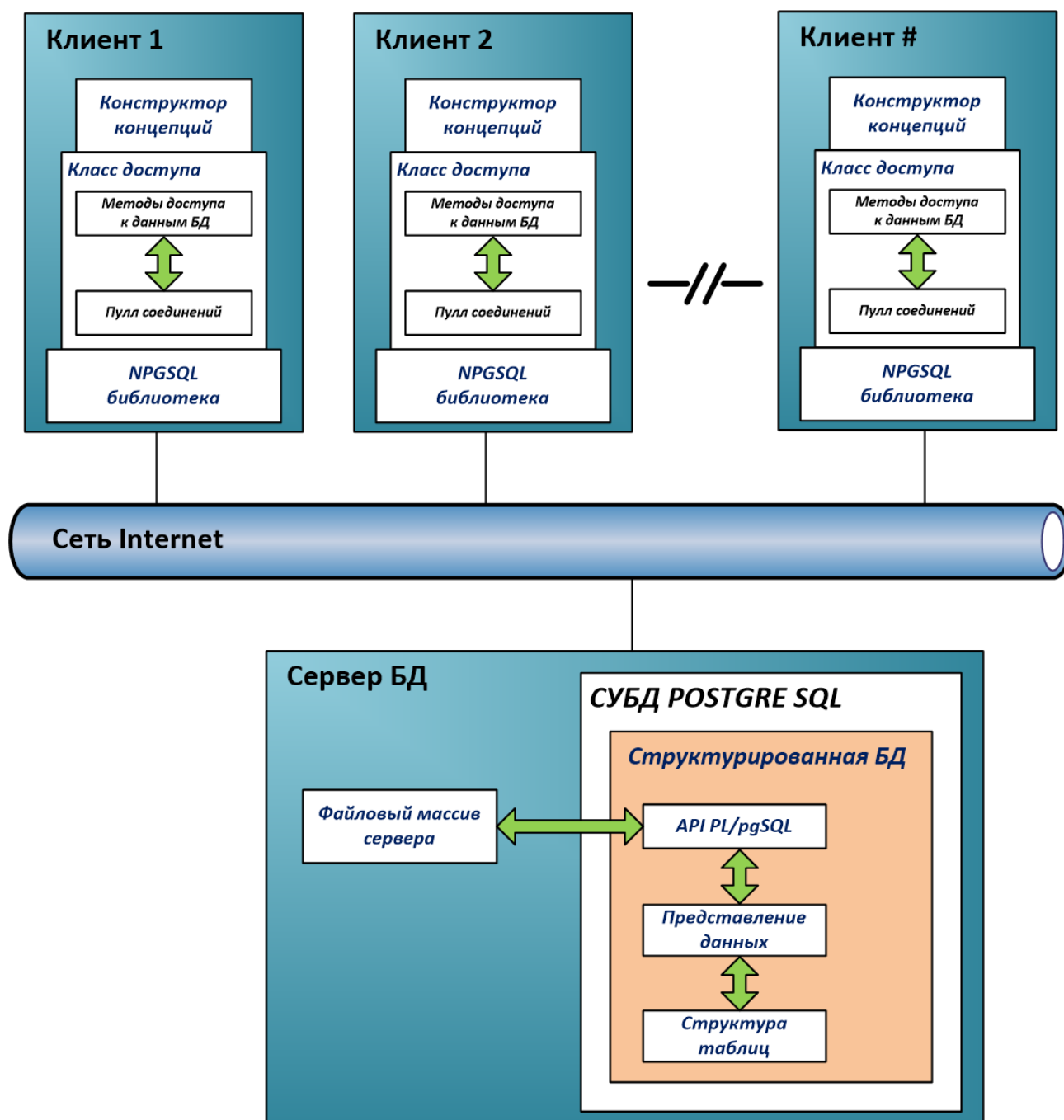
Конструктор концепций описание

№	Сущность	ID сущности	Событие/Состояние	Результат	Код результата	Дата	Время
19	Менеджер данных	0	Инициализация	Пул соединений создан	0	18.11.2020	18:52:28.106
18	Сущность базы данных	-1	Выполнение хранимой п...	Выполнение команды: map_get_baseid с параметрами: ; Время выполнения: 1,001мс	0	18.11.2020	18:52:28.087
17	Сущность базы данных	-1	Выполнение хранимой п...	Выполнение команды: con_by_all с параметрами: status = all; ; Время выполнения: 0,999мс	0	18.11.2020	18:52:28.080
16	Сущность базы данных	-1	Выполнение хранимой п...	Выполнение команды: pos_prototype_nested_by_id_prototype с параметрами: id_prototype = 0; ; Время в...	0	18.11.2020	18:52:28.076
15	Сущность базы данных	-1	Выполнение хранимой п...	Выполнение команды: usr_roles_by_all с параметрами: ; Время выполнения: 1,0016мс	0	18.11.2020	18:52:28.068
14	Сущность базы данных	-1	Выполнение хранимой п...	Выполнение команды: con_by_all с параметрами: status = all; ; Время выполнения: 2,9992мс	0	18.11.2020	18:52:28.059

Конструктор концепций - представляет собой двухуровневый клиент-серверный, многопользовательский программный комплекс, обеспечивающий управление структурированной базой данных, посредством predefined программного интерфейса - API БД. Структурированная БД обеспечивает возможность конструирования древовидных иерархических структур - Концепций, создаваемых архитекторами (проектировщиками), систематизирующими данные произвольной предметной области в соответствии с принципами объектно-ориентированного подхода. Конструктор концепций обеспечивает среду для существования, модификации и взаимодействия с пользователями не ограниченному количеству Концепций на протяжении всего жизненного цикла. Predefined архитектура индексированного табличного пространства структурированной БД, совместно с API БД обеспечивает высокий уровень производительности для процессов доступа к данным Концепций.

В качестве СУБД используется PostgreSQL. Комплекс состоит из следующих архитектурных элементов:

1. **Структурированная БД** - база данных, созданная в СУБД PostgreSQL (не содержит начальных данных);
2. **Промежуточный API БД** - пакет функций, сформированный преимущественно на языке PL/pgSQL, обеспечивающий ведение структурированного каталога БД, сохранение целостности данных в рамках принятой идеологии, с учетом действующих ограничений и правил конкретного архитектурного решения, применяемого в отдельно взятой Концепции;
3. **Библиотека доступа (Менеджер данных)** - программная оболочка клиентского приложения написанная на С# и обеспечивающая решение вопросов сетевого взаимодействия с сервером СУБД. Для построения менеджера данных использована библиотека NPGSQL. Менеджер данных прозрачно обеспечивает полностью асинхронное взаимодействие с БД, посредством внутреннего пула соединений, обеспечивающего сохранение контекста команд в асинхронном режиме;
4. **Конструктор концепций** - графический, пользовательский интерфейс, написанный на С# и обеспечивающий возможности визуального конструирования для архитектора концепций в соответствии с требованиями предметной области.



Назначение

Конструктор концепций предназначен для ручного или автоматизированного конструирования объектных иерархических структур - Концепций, систематизирующих данные произвольной предметной области в соответствии с принципами объектно-ориентированного подхода, в рамках выработанной архитектором (проектировщиком) идеологии или схемы классификации.

Основываясь на общепринятых принципах ООП архитектор определяет критерии систематизации предметной области, основные взаимодействующие сущности, порядок их взаимного расположения и иерархию отношений, после определения базовых принципов описания предметной области в конструкторе концепций создается требуемая структура - Концепция, при этом принятые архитектором правила взаимодействия элементов Концепции, заданные на этапе конструирования становятся жесткими правилами организации данных, идеологическая целостность которых обеспечивается на уровне API БД.

Простой пример:

Рассмотрим элементарный пример, коробок спичек. Для реализации коробка спичек необходимо определить две ключевых сущности — это сам коробок и, собственно, спички. Для коробка так же необходимо определить критерий вместимости, количество спичек, которое в него возможно поместить и, если мы не желаем использовать коробок не по назначению необходимо указать что коробок может содержать только спички. Определив указанные правила в конструкторе концепций, мы получим объект коробок, который может содержать указанное количество спичек и ничего более. Таким образом конечный пользователь осуществляющий ввод и модификацию данных в рамках действующей концепции не сможет обойти правила, жестко заданные архитектором на этапе визуального конструирования.

Конструктор концепций инкапсулирует сложность разработки БД в многопользовательской среде, скрывает вопросы открытия и поддержания сессий, взаимодействия с табличными пространствами, снижая тем самым интеллектуальный порог вхождения в область создания и ведения сложных многопользовательских БД. Архитектор Концепции, должен владеть базовыми знаниями ООП и быть экспертом в систематизируемой предметной области. Этих навыков достаточно для создания Концепции любой сложности. Работа в конфигураторе концепции позволяет систематизировать собственные знания эксперта в предметной области и охватить весь круг вопросов, следуя по наикратчайшему пути реализации задуманного решения. Многопользовательская среда Конструктора обеспечивает возможность одновременной работы нескольких пользователей над созданием одной Концепции, систематизирующей данные предметной области, что существенно повышает качество конечного результата.

В последствии при необходимости, для реализации целевого графического интерфейса Концепции могут быть привлечены профессиональные программисты. При этом извечный вопрос “чего все-таки хотел пользователь?” на данном этапе будет иметь достаточно качественный и вполне осязаемый ответ.

Основные принципы

Создание базы данных, обеспечивающей хранение данных в соответствии с требованиями определенной предметной области с учетом всех возможных аспектов и тонкостей было и остается достаточно сложной задачей. При необходимости обеспечить возможность расширения и модификации системы хранения, подразумевающую внесение заранее не определенных изменений, возникающих в процессе эксплуатации БД и вызванных изменениями в самой предметной области или изменениями требований к структуре представления данных, делают эту задачу еще более сложной. При необходимости сохранения исторической ретроспективы, обеспечивающей преемственность данных, существующих в различных временных срезах и имеющих различия в структурах представления делает данную задачу практически не выполнимой.

По мнению авторов изменение требований к структурам БД является основным фактором, ограничивающим жизненный цикл решений и в тоже время двигателем для дальнейшего совершенствования способов представления данных, обеспечивающих создание новых, более гибких и приспособленных к быстро изменяющимся реалиям вариантов структурирования данных. Один из таких вариантов представлен в настоящем решении.

При создании конструктора концепций упор был сделан не на реализацию структуры хранения данных под определенную предметную область, а на создания системы хранения и модификации заранее неопределенных структур хранения данных, обеспечивающей их создание, модификацию, наполнение и ведение на протяжении всего жизненного цикла. Таким образом конструктор концепций находится на более высоком уровне абстракции по отношению к понятию Концепции и представляет собой среду, обеспечивающую существование неопределенного количества Концепций, на протяжении всего жизненного цикла, независимо от их внутреннего устройства.

Для обеспечения структурной целостности Концепций в среде реализован механизм хранения и поддержания правил организации данных на основе требований архитектора, для взаимодействия с Концепциями был разработан программный интерфейс (написанный преимущественно на языке PL/pgSQL), обеспечивающий все виды необходимого взаимодействия с элементами отдельных Концепций, в том числе: внесение, изменение, удаление данных, поиск, навигация и выполнение математических операция над родственными юнитами.







Основные понятия

Для детального описания принципов работы Комплекса необходимо пояснить основные используемые понятия. Все сущности Конструктора концепция условно разделены на организационные юниты и объектные юниты.

Организационные юниты

Организационные юниты используются для создания древовидных структур, определяющих общую схему Концепции, и имеют трехуровневую архитектуру, состоящую из следующих компонентов:

Полный список прототипов позиций

Прототип (ранг)	Параметры вложений				Сервисный
	Корневой	Вложение в самого себя	Вложение позиций	Вложение объектов	
 RANK:0 ROOT:1 THIS:0 POS:1 OBJ:1	✓	×	✓	✓	✓
 RANK:1 ROOT:1 THIS:0 POS:1 OBJ:0	✓	×	✓	×	×
 RANK:2 ROOT:1 THIS:1 POS:1 OBJ:0	✓	✓	✓	×	×
 RANK:3 ROOT:0 THIS:1 POS:1 OBJ:1	×	✓	✓	✓	×
 RANK:4 ROOT:0 THIS:0 POS:1 OBJ:1	×	×	✓	✓	×
 RANK:5 ROOT:0 THIS:0 POS:0 OBJ:1	×	×	×	✓	×

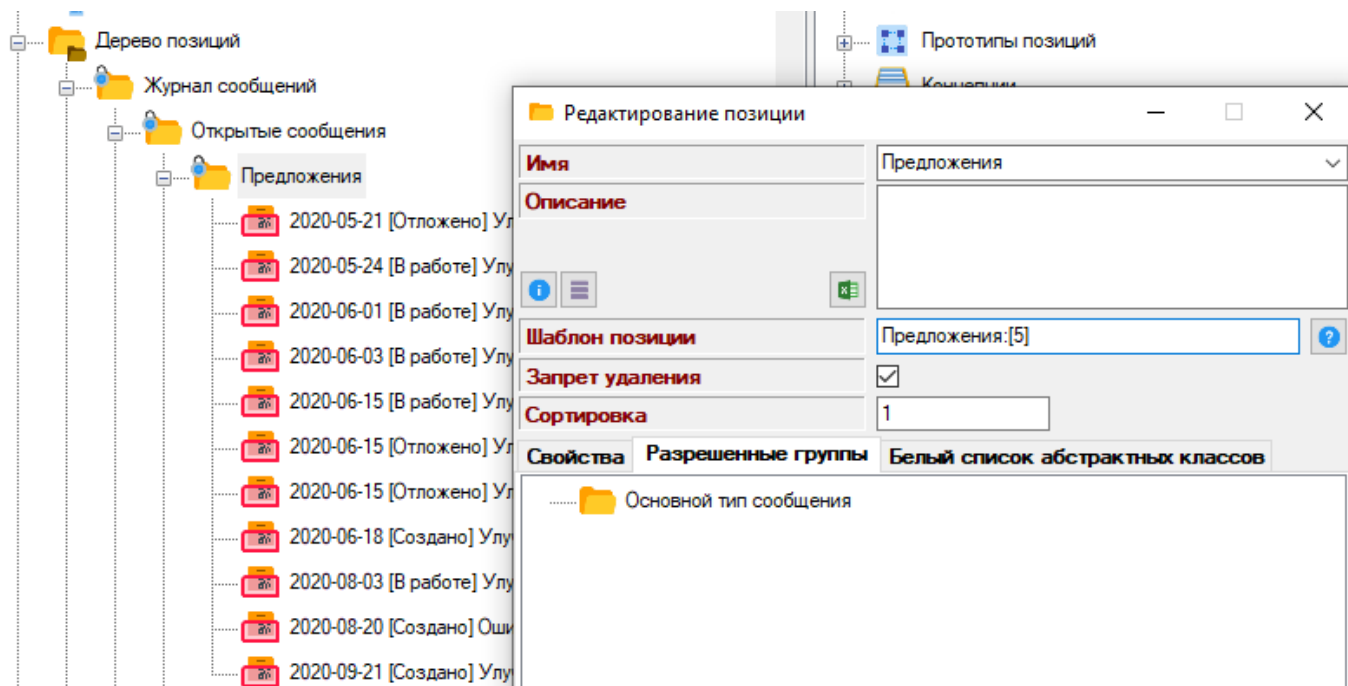
Учет 2.0 - Конструктор концепций

[Визуализация](#)
[Сервисные операции](#)
[Импорт и экспорт](#)
[Инструменты](#)

Объектная среда

- Пользователи
- Роли
- Прототипы позиций
 - RANK:1 ROOT:1 THIS:0 POS:1 OBJ:0
 - RANK:2 ROOT:1 THIS:1 POS:1 OBJ:0
 - RANK:2 ROOT:1 THIS:1 POS:1 OBJ:0
 - RANK:3 ROOT:0 THIS:1 POS:1 OBJ:1
 - RANK:3 ROOT:0 THIS:1 POS:1 OBJ:1
 - RANK:4 ROOT:0 THIS:0 POS:1 OBJ:1
 - RANK:5 ROOT:0 THIS:0 POS:0 OBJ:1

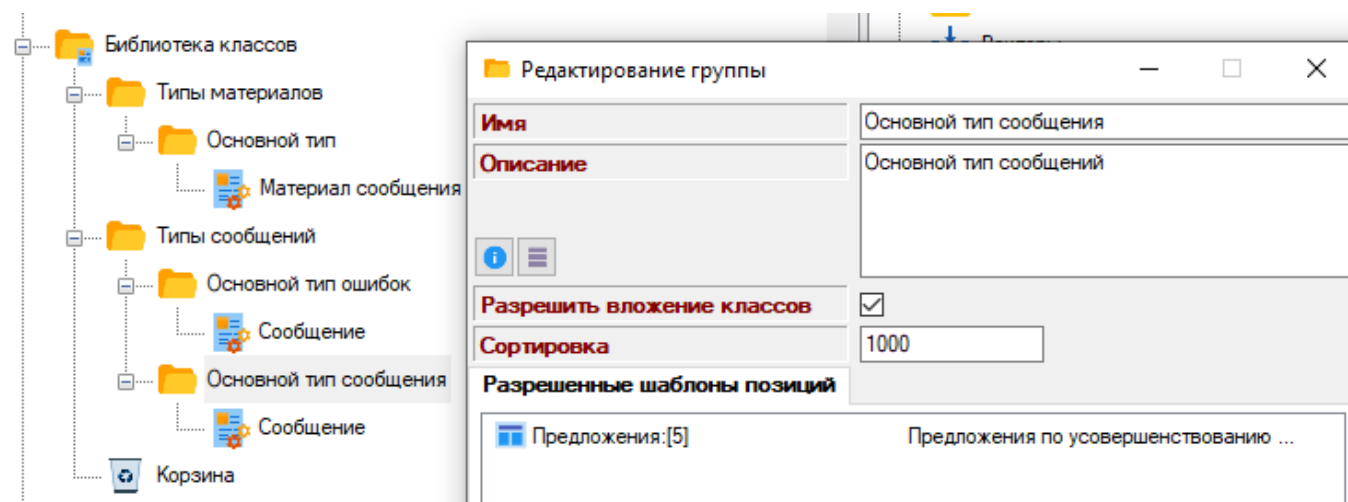
Позиции — это иерархически структурированные примитивы, непосредственно наследующие от шаблонов позиций в части модели иерархического поведения и свойств шаблонов позиций. Позиции могут непосредственно содержать объекты или включать их в качестве значений объектных свойств.



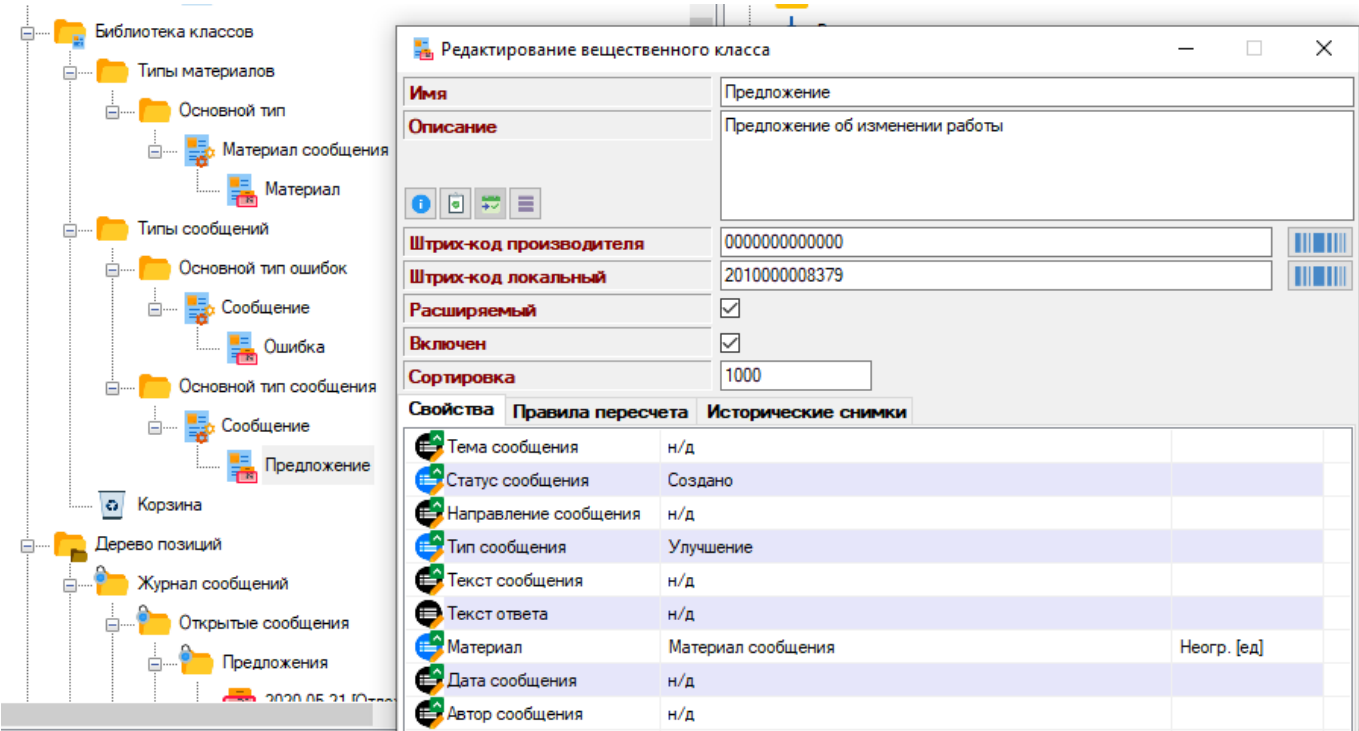
Объектные юниты

Объектные юниты являются основными носителями данных Концепции и имеют трехуровневую архитектуру, состоящую из следующих компонентов:

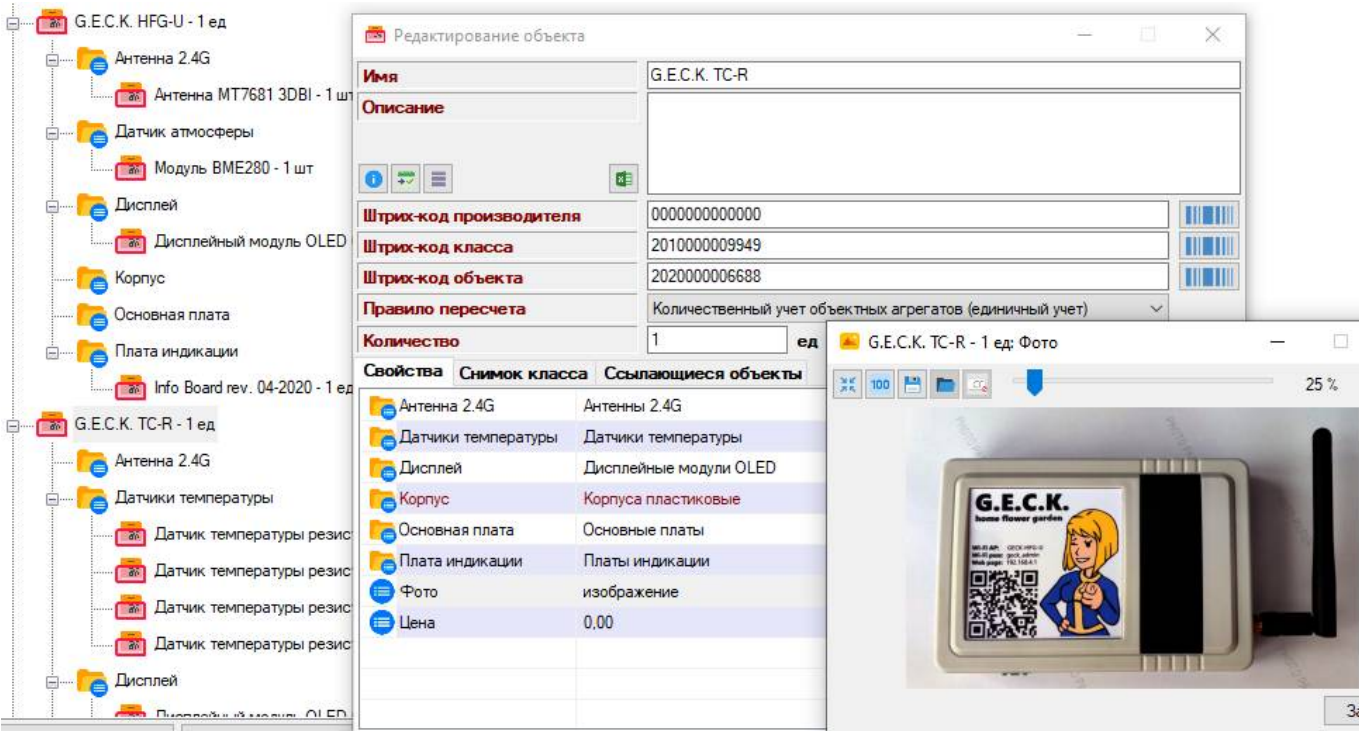
Группы — это иерархически структурированные примитивы, определяющие организационную структуру классов Концепции и правила вложенности первого уровня, **Группа - Шаблон**, для Позиций допускающих вложение объектов.



Классы — это иерархически структурированные примитивы, создаваемые в конечных группах, не содержащих другие группы. Класс, непосредственно содержащийся в группе, считается базовым Классом. Класс, содержащий другие Классы, считается абстрактным. Класс, порождающий Объекты, считается вещественным. В классах определяются правила вложенности второго уровня, **Класс - Позиция**, для Позиций допускающих вложение объектов. В классах могут определяться дополнительные строго типизированные произвольные свойства объектных юнитов.



Объекты — это иерархически структурированные (на основе объектных свойств объектов) примитивы, непосредственно наследующие от классов и создаваемые в Позициях допускающих вложение объектов или в объектных свойствах объектов или позиций. Объект является основным и наиболее массовым носителем данных в Концепции.



Правила вложенности

В основе поддержания организационной структуры Концепции лежат правила вложенности юнитов, определяющие общий замысел архитектора и ограничивающие возможности иерархического взаимодействия юнитов до пределов, обусловленных архитектурным решением.

Правила вложенности организационных юнитов

Правила вложенности организационных юнитов имеют двухуровневую архитектуру:

1. **Уровень прототипа позиции** - обусловлен выбранным прототипом шаблона позиции, а точнее его свойствами on_position и on_this, определяющими в принципе возможность вложения позиций в позицию шаблона, определенного на основе конкретного прототипа;

Полный список прототипов позиций						Параметры вложений				Сервисный	
Прототип (ранг)						Корневой	Вложение в самого себя	Вложение позиций	Вложение объектов		
	RANK:0	ROOT:1	THIS:0	POS:1	OBJ:1	✓	×	✓	✓	✓	
	RANK:1	ROOT:1	THIS:0	POS:1	OBJ:0	✓	×	✓	×	×	
	RANK:2	ROOT:1	THIS:1	POS:1	OBJ:0	✓	✓	✓	×	×	
	RANK:3	ROOT:0	THIS:1	POS:1	OBJ:1	×	✓	✓	✓	×	
	RANK:4	ROOT:0	THIS:0	POS:1	OBJ:1	×	×	✓	✓	×	
	RANK:5	ROOT:0	THIS:0	POS:0	OBJ:1	×	×	×	✓	×	

2. **Уровень шаблона позиции** - обусловлен белым списком шаблона позиций, определяющим возможность вложения позиций указанных шаблонов в позиции настраиваемого шаблона;

Редактирование шаблона позиции		—	□	×
Имя	Открытые			
Описание	Открытые сообщения находящиеся в работе			
Включен	<input checked="" type="checkbox"/>			
Сортировка	1			
Прототип позиции	Ограничение вложения	Разрешенные группы	Свойства	
Включено	<input type="checkbox"/>			
<input checked="" type="checkbox"/> Ошибки:[5]	Неогр.			
<input checked="" type="checkbox"/> Предложения:[5]	Неогр.			

Правила вложенности объектных юнитов

Правила вложенности объектных юнитов имеют трехуровневую архитектуру:

1. **Уровень прототипа позиции** - обусловлен выбранным прототипом шаблона позиции, а точнее его свойством `on_object`, определяющим в принципе возможность вложения объектов в позицию шаблона;
2. **Уровень шаблона позиции** - обусловлен списком правил вложенности объектов **“Группа - Шаблон”**, определяющим возможность вложения объектов классов выбранных групп в позиции настраиваемого шаблона;

The screenshot shows a window titled "Редактирование шаблона позиции" (Editing position template). It contains several fields and controls:

- Имя** (Name): A text field containing "Ошибки" (Errors).
- Описание** (Description): A text area containing "Ошибки в работе базы и клиента" (Errors in database and client work).
- Включен** (Enabled): A checkbox that is checked.
- Сортировка** (Sorting): A text field containing "1000".
- Прототип позиции** (Position prototype): A tab selected among "Прототип позиции", "Ограничение вложения" (Limiting nesting), "Разрешенные группы" (Allowed groups), and "Свойства" (Properties).
- Способ отображения** (Display method): Radio buttons for "Лист" (Leaf) and "Дерево" (Tree), with "Лист" selected.
- Основной тип ошибок** (Main error type): A yellow folder icon.

3. **Уровень позиции** - обусловлен списком правил вложенности объектов **“Класс- Позиция”**, определяющим возможность вложения объектов классов в позиции;

Иерархия правил вложенности имеет уточняющий характер, сложение правил осуществляется по “И”, правила уровней 1 и 2 являются обязательными для разрешения действия, правила уровня 3 носят ограничительный характер, при их отсутствии разрешение определяется сложением правил уровней 1 и 2.

Свойства

В шаблонах позиций и классах могут произвольно определяться дополнительные строго типизированные свойства. Свойства могут быть определены как переопределяемые или не переопределяемые по значению в наследующих юнитах. Свойства шаблонов наследуются позициями. Свойства классов наследуются наследующими классами и порожденными от них объектами.

Редактирование вещественного класса

Имя

G.E.C.K. HFG-U

Описание

Штрих-код производителя

0000000000000

Штрих-код локальный

2010000006214

Расширяемый

☒

Включен

☒

Сортировка

1000

Свойства

Правила пересчета

Исторические снимки

Антенна 2.4G	Антенны 2.4G	0 - 1 [шт]	
Датчик атмосферы	Модули атмосферных датчиков	1 - 1 [шт]	
Дисплей	Дисплейные модули OLED	1 - 1 [шт]	
Корпус	Корпуса пластиковые	1 - 1 [шт]	
Основная плата	Основные платы	1 - 1 [ед]	
Плата индикации	Платы индикации	1 - 1 [ед]	
Фото	изображение		
Цена	3500,00		

Применить

Заккрыть

Простой пример:

При описании устройства полезно определить не переопределяемое по значению в объектах свойство “Модель” и переопределяемое свойство “Серийный номер”, в данном случае конечный пользователь, вносящий данные в БД, не сможет изменить данные о модели конкретного устройства, но сможет указать ее индивидуальный серийный номер.

Все свойства разделены на четыре основных типа:

- **Пользовательское** - строго типизированное свойство с областью значений, определяемой выбранным типом данных свойства;
- **Перечисление** - строго типизированное свойство с областью значений, определяемой предварительно созданным пользовательским перечислением;
- **Объектное** - строго типизированное свойство с областью значений, определяемой выбранным абстрактным классом, в качестве значений объектного свойства выступают объекты классов, входящие в ветвь выбранного абстрактного класса - области значений. Объект, содержащий объектные свойства, считается **Объектным агрегатом** и подлежит исключительно единичному количественному учету;

- **Ссылка** - строго типизированное свойство с областью значений, определяемой выбранной сущностью БД, в качестве значений ссылочного свойства выступают примитивы БД, относящиеся к указанной сущности. В настоящий момент доступны для связывания: пользователи, шаблоны позиций, свойства шаблонов позиций, позиции, свойства позиций, группы, классы, свойства классов, объекты и свойства объектов;

Глобальные свойства

Применение механизма наследования свойств позволяет автоматизировать работу со свойствами наследующих юнитов, однако разрозненные группы классов, содержащих идеологически родственные свойства, не могут быть автоматически сопоставлены в данном случае, в виду отсутствия четких критериев для связывания. Кроме того свойства определенные в позициях и объектах так же не могут быть сопоставлены между собой с использованием только механизма наследования. Для решения этой задачи были созданы **глобальные свойства**, назначение которых состоит в определении совместимости разных свойств и их последующей линковки. Все свойства сопоставленные с глобальным должны иметь одинаковое смысловое значение, совпадать по имени, описанию, типу свойства, области значений и типу данных.

С глобальными свойствами в зависимости от их типа, области значений и типа данных возможно выполнение общих операций таких как математические действия, поиск и фильтрация по маске значения и т.п.

Поиск

Концепция для поиска

Мастерская

Позиция поиска

Кладовая

Сущность для поиска

Объекты

Поле поиска

Значение свойства глобального

Свойство глобальное

Производитель

Ключ поиска

Microchip

Производительность

Получение данных: 00:00:01.393

Формирование списка: 00:00:00.057

Найдено: 2

Объект	Произво...	Он...	Кол...	Путь
Микроконтроллер ATMEGA328-AU	Microchip		5 шт	Кладовая\Стеллаж\Контейнер N°2\A02
Микроконтроллер ATMEGA328-AU	Microchip		5 шт	Кладовая\Стеллаж\Контейнер N°2\A02

Поиск

Закрыть

Куб метаданных

Простота внесения изменений в существующие структуры данных порождает множество мелких изменений, вносимых архитекторами в структуру готовой Концепции. Для обеспечения исторической структурной целостности данных был создан “Куб метаданных” хранящий всё используемое существующими объектами множество мгновенных состояний классов Концепции - **снимков классов**. Каждый не удаленный из Концепции класс представлен активным состоянием (текущее состояние класса) используемым для порождения новых объектов и совокупностью мгновенных снимков, созданных при порождении существующих объектов. Таким образом все объекты несмотря на свои видимые различия оказывается связаны родственным классом и опираются каждый на свое мгновенное состояние. При необходимости любой объект может приведен к любому доступному состоянию класса.

Имя	Описание	Штрих-код производителя	Штрих-код класса	Штрих-код объекта	Правило пересчета	Количество
G.E.C.K. HFG-U		0000000000000	2010000006214	2020000012948	Количественный учет объектных агрегатов (единичный учет)	1 ед

Свойства	Снимок класса	Ссылающиеся объекты
G.E.C.K. HFG-U	Информация	29.09.2020 6:56:53.549
G.E.C.K. HFG-U	Обновить	29.09.2020 9:50:36.590
G.E.C.K. HFG-U1	Привести к снимку	10.10.2020 9:17:14.409
G.E.C.K. HFG-U	Показать объекты	10.10.2020 9:19:34.042
G.E.C.K. HFG-U1		10.10.2020 9:58:50.269
G.E.C.K. HFG-U		10.10.2020 10:00:30.692
G.E.C.K. HFG-U1		10.10.2020 10:07:52.638
G.E.C.K. HFG-U		10.10.2020 10:08:10.604
G.E.C.K. HFG-U		Активный класс

Таким образом, использование массива опорных снимков классов для поддержания состояния объектов позволяет разделить процессы модификации классов и наследования этих модификаций ранее созданными объектами, что обеспечивает сохранение исторической ретроспективы данных в структуре Концепции.

Простой пример:

Рассмотрим элементарный пример, изготовитель продукции изменил маркировку товара, при этом один и тот же по сути товар оказался с двумя маркировками до и после. Соответственно если просто изменить маркировку товара в классе, то все ранее внесенные товары так же получат данное изменение что может негативно сказаться на фискальной отчетности, возможности сопоставления печатных отчетов или заверенных документов, накладных и т.п. Если завести новый класс при наличии товара в виде складских запасов со старой маркировкой так же возникнут неудобства при сопоставлении экземпляров. “Куб метаданных” позволит мирно сосуществовать товарам одного вида с разными маркировками без каких либо, действий со стороны архитектора концепции, так как в данном случае каждый объект будет использовать свой снимок данных, содержащий сведения о маркировке, актуальные на момент создания объекта.

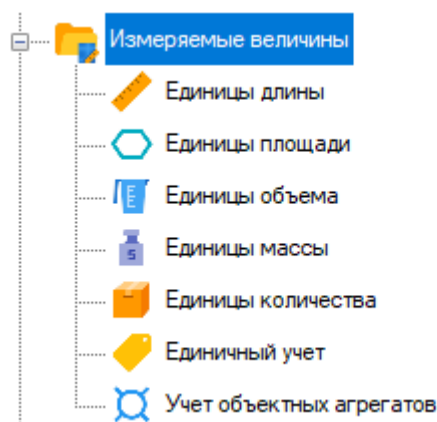
Библиотека документов

Гибкая возможность реорганизации данных обеспечивает широкие возможности для сохранения сложной, многогранной информации, однако хранение информации пригодной для размещения в табличном пространстве БД существенно ограничивает возможности Концепций. Для полноценной Концепции необходимо файловое хранилище, обеспечивающее структурированное хранение электронных версий документов, связанных с данными Концепции. Существует множество вариантов агрегации СУБД и файловых серверов, однако все они требуют развертывания, настройки и администрирования дополнительного программного обеспечения, усложняющего сопровождение Комплекса. Использование системы хранения больших объектов СУБД PostgreSQL так же имеет ряд недостатков, ограничивающих возможности и усложняющих планы резервного копирования, в том числе из-за увеличения размеров каталога СУБД. При создании каталога документов был разработан третий, промежуточный вариант на основе виртуальных таблиц, формируемых функциями API БД. Электронный документ при размещении в файловом каталоге обслуживаемом API БД, на стороне клиента преобразуется в байтовый массив и отправляется на сервер как значение ячейки таблицы `bytea`, при попадании на уровень API БД сервера байтовый массив перехватывается соответствующей функцией и сохраняется в структурированный файловый каталог за пределами каталога СУБД. Запрос файлов происходит в обратном направлении. Таким образом файловый обмен реализован без использования каталога СУБД и стороннего программного обеспечения, штатными функциями обмена данными СУБД PostgreSQL, что существенно упрощает программирование файлового обмена в целевом программном интерфейсе готовой Концепции.

Служебные дополнения

Правила пересчета объектов

Количественный учет данных, одна из ключевых задач любой БД. Для решения этой задачи в структурированной БД Конструктора концепций определены допустимые измеряемые величины:



Список правил пересчета

Концепция	Мастерская						
Измеряемая величина	Единицы количества						
Правила пересчета измеряемой величины							
Наименование	Сокращение	Базовая ед. изм.	Козф. пересчета	Округление	Используется	Ед. учет	Базовое
штука	шт	шт	1	0	✓	x	✓
упаковка по 5 штук	уп x 5 шт	шт	5	0	x	x	x
упаковка по 10 штук	уп x 10 шт	шт	10	0	x	x	x
упаковка по 20 штук	уп x 20 шт	шт	20	0	✓	x	x

Пользовательские строго типизированные свойства практически полностью перекрывают потребность в создании определяемых пользователем свойств. Однако существуют ситуации, в которых требуется дополнительная формализация значений свойств до значений, выбираемых из выпадающего списка. Существует возможность поддерживать такие списки в ресурсах целевых клиентских приложений, однако это накладывает ряд существенных ограничений на клиентские приложения и усложняет стратегии их развертывания и обновления. Для преодоления вышеописанных недостатков Конструктор концепций предоставляет возможность создания перечислений и назначения существующих перечислений в качестве области значений свойств типа перечисление.

Скриншот диалогового окна «Редактирование перечисления» в программе «Учет». В окне отображены следующие данные:

- Имя:** Типы диодов
- Описание:** (пустое поле)
- Тип:** Строковый тип
- Область применения:** Объекты
- Значения перечисления:**

Значение	Используется
* Выпрямительный	×
* Высоковольтный	×
* Защитный	×
* Импульсный	✓
* СВЧ	×
* Силовой	×
* Стабилитрон	×
* Шоттки	×

В нижней части диалогового окна расположены кнопки «Применить» и «Закрыть».

Экспорт данных в Excel

Для реализации задачи экспорта данных используется библиотека python `openpyxl` задействованная в процедурах экспорта табличных данных на стороне сервера. Готовый двоичный файл Excel преобразуется в байтовый массив и передается клиенту как элемент виртуальной таблицы, на стороне сервера полученные данные сохраняются в файл экспорта. Использование данного метода существенно снижает требования к клиентскому хосту так как не требует наличие пакета офиса или установленных сторонних библиотек необходимых для обработки файлов Excel

The screenshot displays a web application interface on the left and a LibreOffice Calc spreadsheet on the right. The web application shows a hierarchical tree structure under 'Мастерская' (Workshop) with folders like 'Монтажный стол' (Assembly table) and 'Сборка' (Assembly). Under 'Сборка', there is a folder 'G.E.C.K. HFG-U - 1 ед' (G.E.C.K. HFG-U - 1 unit) containing components like 'Антенна 2.4G' (2.4G Antenna), 'Датчик атмосферы' (Atmosphere sensor), 'Дисплей' (Display), 'Корпус' (Case), 'Основная плата' (Main board), and 'Плата индикации' (Indicator board). Below the tree, there are buttons 'Обновить (F2)' (Refresh) and 'Открыть в новом окне' (Open in new window). The user is 'Иванов Д.Ю.' (Ivanov D.Yu.) and the database is 'Uchet [DB build]'. The 'Журнал событий' (Event log) shows a list of events with columns '№' (Number), 'Сущность' (Entity), and 'ID сущности' (Entity ID).

The LibreOffice Calc spreadsheet, titled 'G.E.C.K. HFG-U.xlsx', shows a table with the following data:

	L	M	N	O
1	name	id_class	root	id_group
2	Info Board rev. 04-2020	591	206	206
3	Антенна MT7681 3DBI	500	141	141
4	Дисплейный модуль QLED 0,91	313	141	141
5	Модуль BME280	313	141	141
6	Резистор SMD-0805 110 Ом 1%	400	138	138
7	Светодиод SMD-1206 белый Vanxy	-1	138	138
8	Светодиод SMD-1206 желтый Vanxy	-1	138	138
9	Светодиод SMD-1206 зеленый Vanxy	-1	138	138
10	Светодиод SMD-1206 красный Vanxy	-1	138	138
11	Светодиод SMD-1206 синий Vanxy	-1	138	138

Описание депонируемых материалов

Депонируемые материалы состоят из пяти основных разделов:

1. Графическое представление структурированного каталога БД, пакет схем;
2. Исходные коды функций API БД на PL/pgSQL;
3. Исходные коды клиентской библиотеки доступа (Менеджера данных), написанные на C#;
4. Графическое представление экранных форм клиентского интерфейса (Конфигуратора концепций);
5. Исходные коды графического клиентского интерфейса (Конфигуратора концепций), написанные на C#.

Список используемого стороннего ПО

При создании Конструктора концепций использовано ПО сторонних авторов, распространяемое под общественной лицензией GNU или либеральными лицензиями для ПО с открытым исходным кодом, аналогичные лицензиям BSD или MIT.

Наименование ПО	Тип лицензирования
СУБД PostgreSQL	Либеральная лицензия с открытым исходным кодом, аналогичная лицензиям BSD или MIT
Библиотека доступа для платформы .Net Npgsql	Находится под лицензией PostgreSQL , либеральной лицензией с открытым исходным кодом, одобренной OSI.
Язык программирования Python	Python Software Foundation License (PSFL) — BSD-подобная перmissive лицензия на свободное ПО, совместимая с GNU General Public License (GPL).
.NET Framework	Регламентируется условиями лицензии ОС MS Windows.
openpyxl - библиотека Python	Лицензия MIT / Expat