

My4S – Програма-конструктор для створення web-застосунків, таких як ERP, CRM (Написання програм схожих на 1C8.x, BAS та інших)

Автор: Serhiy Chakhanyuk aka didsergiy4s.

Зміст

Що таке My4S.....	4
Основні принципи:.....	5
Технології:	5
Коротко про My4S	5
На клієнті:.....	5
На сервері:	5
ODBC драйвер.....	6
Особливості My4S.	6
Що маємо в підсумку:.....	7
Форми браузера	7
Оптимізація роботи з базою даних	7
Можливості My4S.....	8
Програма підтримує:	8
Майбутнє My4S	8
Перспективи	8
Механізм відновлення бази даних.....	8
AUTHORIZATION	9
Щоб замінити пароль:	9
Управління правами доступу в Підприємстві	10
Зміна пароля.....	10
Зв'язок з користувачем	10
Початкова база(не путайте з demo).....	11
Конструктор.	11

Таблиця METADATA та структура меню.....	11
Додавання нових об'єктів.....	11
Для початку подивимся ID_DB.....	11
Основні ідентифікатори метаданих.....	11
Обмеження на створення ідентифікаторів	12
Додаткові властивості типів date, select i const.....	12
BALANSE (Регістри залишків):	12
Залишки в My4S та їх підтримка у Firebird	12
Чому accounts і accounts#1.....	15
Робота форми з налаштування метаданих.....	16
Комплекти.....	19
Підпорядкований довідник (CATS1).....	20
Регістри відомостей (REGS).....	20
Документи (DOCS).....	21
Редагування текстового поля (textarea)	22
Колонка «ADD Attributes»	22
Вкладка «DESCRIPTION».....	23
Нове (Excel)!	23
Опис рухів регістрів	24
Меню BALANSE	25
Введення символу для руху	25
Автоматичне формування рухів	26
🔧 Робота механізму створення документа на підставі в системі «Підприємство».....	27
Реалізація та роль програміста	28
■ Перевірка документа при відкритті	28
Перевірка функції MyFunctBASIS.....	28
Функція GET_BLOB у Firebird.....	29
Структура 4SON.....	29
Функція GET_BLOB у основній програмі (JS).....	30
Про знання 4SON.....	31

■ Опис структури файлів проекту на Node.js.....	31
📁 .vscode	31
📁 Структура папок проекту.....	32
📁 Структура папок бази даних.....	32
📁 Внутрішні папки бази	33
📁 Папка db – шаблони .htm та інші файли для роботи в Підприємстві.....	33
Як у мене працює src:	34
Звіти в My4s	34
■ Вибір регистра.....	35
Приклади роботи з відборами	36
А тепер опишемо, як працює Settings.....	37
■ Як зберігаються налаштування	38
Звіти, створювані в Конструкторі	38
Створення нового звіту	39
Загальні принципи налаштування звіту	40
Робота з колонкою AS	40
Додавання нового рядка	40
Параметри Name, VV і TAB.....	40
Приклад використання	41
Закладка MaketPrint:.....	42
Наступний звіт — Помічник написання додаткових звітів.....	43
Початок роботи з новим звітом	43
Звіти query_fast (Швидкі звіти).....	49
Processing (Обробки).....	50
Підсумуємо звіти та обробки.....	50
Firebird select	51
Робота з використанням різних часових зон	52
Підсумуємо вище написане.	52
Керівництво по встановленню My4s на ваш комп'ютер або ноутбук з Windows.....	52

Створення папок і бази.....	52
Копіювання додаткових файлів	52
Встановлення Firebird	52
Встановлення ODBC драйвера	53
Робота з My4s під Windows 7	55
Кроки для налаштування бази	55
Перевірка роботи	55
Зв'язок з автором:	56
Допомога проекту:	56
Ліцензія та використання(MIT).....	56
Що означає ця ліцензія.....	56
Авторські права	56
Відмова від гарантій	56

Що таке My4S.

**My4S — вільно поширюваний web-застосунок з відкритим вихідним кодом.
Серверна частина працює на Node.js (поки що лише під Windows).**

My4S може стати чудовим інструментом для малого та середнього бізнесу, особливо якщо вам потрібна гнучка та безкоштовна ERP/CRM система. Ось кілька способів його використання:

1. Управління фінансами та бухгалтерією.

Ви можете налаштувати реєстри бухгалтерії для обліку доходів, витрат, податків та інших фінансових операцій. Вбудовані оборотно-сальдові відомості допоможуть аналізувати стан бюджету.

2. Облік складських запасів.

Завдяки можливості працювати з різними регістрами та довідниками можна легко вести облік товарів, відстежувати залишки, автоматизувати закупівлі та аналізувати продажі.

3. Управління клієнтами та продажами.

Використовуйте довідники контрагентів і документообіг для ведення клієнтської бази, управління замовленнями, автоматизації виставлення рахунків та аналізу продажів.

4. Автоматизація бізнес-процесів.

My4S дозволяє створювати документи та реєстри, що дає змогу автоматизувати ключові процеси — від розрахунків заробітної плати до планування зустрічей і завдань.

5. Налаштування під будь-які завдання.

Оскільки система гнучка і з відкритим кодом, ви можете адаптувати її під власні потреби: додати нові функції, розширити звітність, інтегрувати з іншими сервісами.

Висловлюю велику вдячність розробникам Node.js, баз даних Firebird, Firebird Editor Pro, Visual Studio Code!

Основні принципи:

- Використання виключно безкоштовного ПЗ (за винятком Windows).
- У Node.js поки використовується лише один ехе-файл, і це принципова позиція! Важливо, щоб сторонній код, використаний у програмі, викликав довіру або міг бути доопрацьований без участі автора.
- Можливість підключення будь-якої кількості зовнішнього коду — як у Node.js, так і в усьому, що підтримує JavaScript. Однак відповідальність за його використання покладається на вас!

Технології:

- **База даних: Firebird 3.0 і вище. Для адміністрування можна використовувати Firebird Editor Pro.**

Редактор: Якщо у вас ще немає улюбленого редактора JavaScript, спробуйте Visual Studio Code — він пропонує непогану підтримку Node.js. Прямо з редактора перезапускається Node.exe, що дуже зручно для внесення змін і тестування. Можна запускати в режимі відладки!

- **Клієнт і сервер:** Обидва пишуться однією мовою — JavaScript.

Коротко про My4S

На клієнті:

Не потрібно встановлювати жодних програм! Використовуємо лише браузер (тестувалося у браузерах Google Chrome, Microsoft Edge, Opera, CCleaner Browser).

Запускається рядком у браузері “path:port/path_mydb/cf” — вхід у Конструктор або “path:port/path_mydb/db” — вхід для роботи з програмою (Підприємство).

Де:

path — шлях до сервера My4S (якщо на локальному комп’ютері — «127.0.0.1», якщо корпоративна мережа — «ім’я сервера з Node». На локальному комп’ютері та корпоративній мережі все працює без інтернету! В іншому випадку — стандартний шлях до сайту або VPN).

port — вільний порт, який буде слухати сервер (прописується у «Start4s.js»). У мене за замовчуванням «1637».

path_mydb — назва папки з конкретною базою (на сервері може бути кілька баз).

Приклад: «127.0.0.1:1637/mydb/cf».

На сервері:

Встановити Firebird 3.x (можна й Firebird 4.x (5.x) — але розробники так погнали галопом!!). Я поки тестував під 3.x, але думаю, що має працювати й на інших (4.x перевірив швидко — працює, 5.x поки сирий!). Для розробки ERP-програм наразі цілком вистачає й 3.x, а далі вирішуйте самі!

Покрокова інструкція, як налаштувати демо-базу та основну базу, наведена наприкінці опису.

ODBC драйвер

Для роботи з Firebird використовую ODBC драйвер. Чому не “node-firebird-driver-native”?

Є кілька причин:

Намагаюся не використовувати код, у якому я не впевнений, що він буде працювати з Firebird x (був у мене досвід з Firebird 2.5, також застосовував драйвер з інтернету — вийшов реліз Firebird 2.8 і драйвер перестав працювати!). Тож тихіше їдеш — далі будеш!

Node.js (однопотоковий) під час роботи використовує одне ядро CPU! Тому я застосовую власний метод роботи з базою Firebird — замість драйвера node-firebird-driver-native використовую DSN ODBC для Firebird і викликаю щоразу асинхронно 'child_process', а потім запускаю функцію, написану та викликану за допомогою "wscript.exe". A Windows автоматично розподіляє процеси між доступними ядрами процесора.

У мене написані два файли під "wscript.exe". Один для запису — "MyComand.js", інший для отримання даних — "MySelect.js". Ці файли написані мовою Jscript під Windows і вони є в дистрибутиві в корені папки "My4S".

Приклад коду:

```
let child_process = require('child_process');

...
let MyWscrComm, MyWscrSel;
bason = require(require('MybasePath').idx(BasonPath));
MyWscrComm = "wscript.exe MyComand.js " + bason.DSN + ' ' + proces;
MyWscrSel = "wscript.exe MySelect.js " + bason.DSN + ' ' + proces;
...
let strExec = MyWscrComm + ' ' + CountLogs[mydbxx]++ + ' ' + mydbxx;
child_process.exec(strExec, () => { ... })
```

Особливості My4S.

Після виходу Firebird 3.0 і вище були суттєво збільшені обмеження на розміри бази даних (до 128 Tb) і таблиць (до 64 Tb). По суті, тепер усе залежить лише від можливостей сервера.

У зв'язку з цим я вирішив робити лише по одній таблиці для документів, довідників, реєстрів відомостей та інших.

Для розуміння розглянемо поки лише одну таблицю — довідники (CATS). А як же в кожну таблицю можна вмістити десятки, а може й сотні різних довідників типу номенклатура, контрагенти і т.д.? У

кожної таблиці є стандартні колонки (`id`, `id_n`, `name`, `actual`, `cod` ...), а для решти колонок будемо використовувати поле `blob(memo)`. І всі інші поля (колонки), унікальні для кожного довідника, запишемо у рядок типу JSON. Думаю, досвідчені програмісти мою ідею вже зрозуміли. NoSQL давно витає у повітря й недарма поважні бази даних почали впроваджувати JSON.

Щоб не бути оригінальним, усі поля, імена, ідентифікатори пишуться англійською мовою та мають довжину до 31 символа.

Що маємо в підсумку:

id — унікальний ідентифікатор таблиці. Тобто, знаючи `id`, ми одразу потрапляємо на елемент будь-якого довідника.

id_n — ім'я довідника, як ви його назовете у Конструкторі (приклад: “`contr`”, “`nomencl`” ...).

`name` — рядок елемента довідника (“Бензин А-95”, “Пиво таке-то”, ...), довжина задається у Конструкторі.

cod — рядок елемента довідника (може містити штрих-код, офіс-код та інші параметри, розділені пробілом або іншим роздільником; довжина задається у Конструкторі).

id_n + name та **id_n + cod** — парні індекси.

Увага! Для довідника «Договори контрагентів» я б рекомендував ім'я «contract». Це не критично, але в деяких випадках програма працюватиме більш точно!

Майже так само все і з реєстром відомостей (REGS), тільки замість `cod` додається поле дата для вибору періодичних даних.

З документами все значно складніше. Використовуватимемо дві пов'язані таблиці: одна для «шапки» (DOCS), а інша для рядків документів (ROWS_DOCS).

Форми браузера

Кожна таблиця (довідники, документи, реєстри відомостей та інші) матиме єдиний шаблон HTML-коду з фіксованими стандартними даними, а також заготовку HTML-таблиці (table) з одним рядком. Для документів передбачено дві таблиці: одна — для «шапки» документа (з одним рядком), інша — для табличної частини (багаторядкова). Така конструкція усуває потребу в редакторі форм (IDE), достатньо використовувати хороший редактор HTML-коду (наприклад, Visual Studio Code).

Перед відкриттям форми шаблон буде динамічно доповнюватися під кожен конкретний об'єкт метаданих, використовуючи дані, записані у `blob`-полях або у JS-файлах.

Оптимізація роботи з базою даних

Програма розроблена таким чином, щоб мінімізувати звернення до бази даних. Усе, що створюється у Конструкторі, описується властивостями кожного метаданого та записується у `blob`-поля таблиці. Одночасно ці дані зберігаються у спеціальний JS-файл, пов'язаний з конкретним метаданим.

Під час відкриття форми метаданого дані не завантажуються з бази, а підключаються через SRC JS-файл. Цей файл оновлюється при кожній зміні метаданих у Конструкторі.

Такі підготовчі JS-файли суттєво скорочують звернення до бази даних, оскільки програма працює з попередньо завантаженими даними.

Можливості My4S

Програма My4S вже готова для створення ERP-систем. Для розробки невеликих застосунків, таких як управління сімейним бюджетом або невеликий складський облік, навіть не потрібні глибокі знання HTML (DOM) чи мов програмування — достатньо ознайомитися з описом My4S.

Програма підтримує:

- реєтри залишків (прості та бухгалтерські, включаючи довідник бухгалтерських рахунків);
- списання собівартості за середнім методом (Fifo та LIFO я поки не описував — можна додати самостійно);
- готові оборотно-сальдові відомості як за реєстрами залишків, так і за реєстрами бухгалтерії (з відборами та збереженням налаштувань);
- до семи багаторядкових таблиць в одному документі.

Майбутнє My4S

Я намагаюся наздогнати можливості 1С8, хоча це нелегко, враховуючи, що система розроблялася понад двадцять років (не рахуючи 1С77). Однак я сподіваюся, що мое починання зацікавить спільноту програмістів-мрійників, які допоможуть перетворити «гідке каченя» на «прекрасного лебедя».

На даний момент я не прагну створювати завершені рішення типу «Бухгалтерія», «УТП», «УПП» та інші. Моя мета — розробити зручний інструмент для створення подібних програм.

My4S — гнучка система, яка дозволяє легко:

- доопрацьовувати наявні модулі та створювати нові,
- створювати нові об'єкти та описувати їхні властивості,
- додавати нові таблиці та функції у базу даних,
- змінювати старі та розробляти нові шаблони HTML.

Перспективи

Загалом My4S — готовий стартап, якому бракує лише одного: щоб хтось повірив у програму.

Механізм відновлення бази даних

Я розробив і успішно протестував механізм відновлення бази даних після критичного збою. Його суть полягає в тому, щоб відновити базу даних точно до моменту збою.

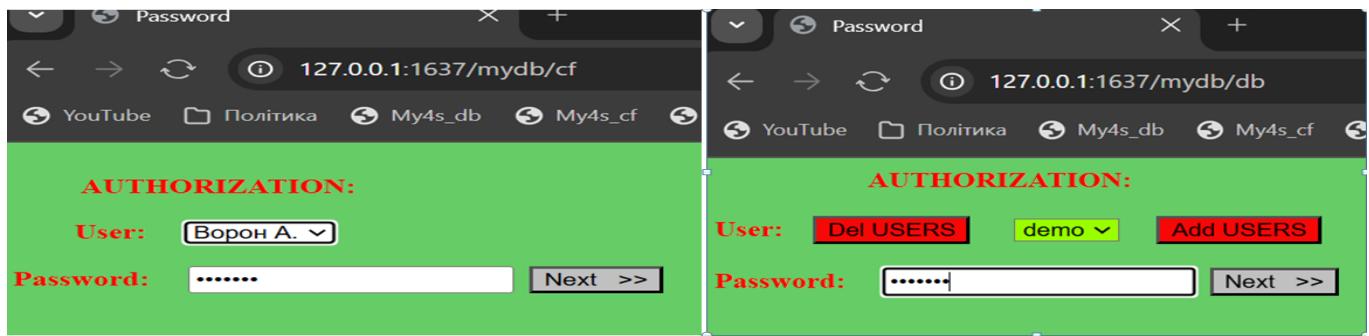
Процес відновлення:

1. Після створення резервного архіву бази даних починається логування всіх операцій UPDATE та INSERT.
2. Усі SQL-запити, що передаються у драйвер Firebird, записуються у файли всередині папки logs.
3. У випадку збою база даних відновлюється з останнього резервного архіву.
4. Далі всі збережені файли у папці logs обробляються у циклі та передаються на виконання у Firebird, таким чином відновлюючи навіть зміни, внесені у Конструктор!

Увага! Поки що я цей механізм не вставляю в My4S. Але в папках logs є папка **bec_res** в якій знаходяться потрібні файли за допомогою яких можна відновити базу. В цих файлах треба правильно прописати шляхи як на вашому сервері а також правильно вказати пароль. ЦЕ буде під силу тільки досвідченим програмістам. Потім я щось придумаю (а може ви підкажете!) щоб було зручно відновляти базу прямо в My4S.

Я не є спеціалістом з безпеки, тому поки не ставив перед собою завдання спеціально захищати ваші дані. Якщо програма виявиться успішною, сподіваюся, знайдуться експерти, які або розроблять надійні рішення, або підкажуть, як їх реалізувати. Поки я знаю лише один спосіб захисту — використання OpenVPN з згенерованими сертифікатами. Однак зараз про це думати зарано, адже для ознайомлення з програмою ви працюватимете на власному комп’ютері й навіть можете обйтися без інтернету.

AUTHORIZATION.



Scr1

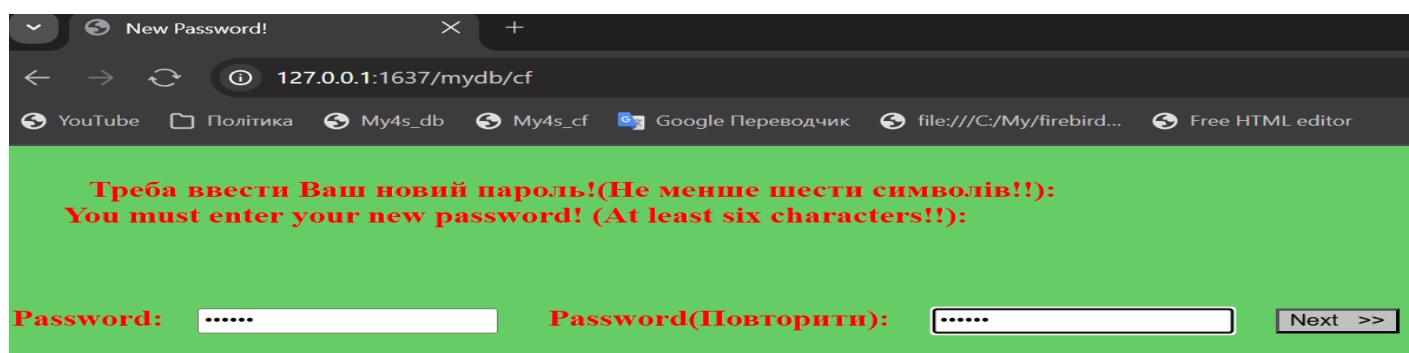
Для входу в Конструктор і в Підприємство для авторизації використовуються два різні підходи.
Для входу в Конструктор дані беруться з файлу JSON “ ..\My4S\mydb\cf\Progvs.us”:

```
{  
  "Prog1": [  
    "Voron A.",  
    "767a06140f52c4cd3f81c9124be765ba038f895644e6d08bb0157c8467b2317b",  
    ""  
  ],  
  "Prog2": [  
    "Denis",  
    "a73317a3d07ff37fa0837338a29e0c9bd6f488b788be6e98bfcd30af9edaf43c",  
    ""  
  ]  
}
```

- **Prog1** — основне ім’я користувача, яке відображається у Підприємстві.
- "Voron A." — представлення, використовується виключно для авторизації.
- Третій елемент — пароль, зашифрований за допомогою модуля Node.js 'crypto'.
- Четвертий елемент — поки не використовується, але в майбутньому може знадобитися для налаштування прав доступу.

Щоб замінити пароль:

- Змініть зашифрований рядок у файлі JSON на тимчасовий пароль (не довший за п’ять символів).
- Увійдіть у Конструктор.
- Введіть новий пароль у вікні, що з’явиться, та підтвердьте його.



Scr2

Новий пароль буде зашифрований і записаний у JSON-файл.

Якщо все пройшло успішно, з’явиться вкладка з інформацією (інакше ERROR):

The new password is attached. Close the tab and go to the program with a new password!

Новий пароль прийнято. Закрийте вкладку і зайдіть у програму вже з новим паролем!

У демо-версії я тимчасово зняв обмеження на шість символів!

За замовчуванням пароль для
«Voron A.» = “11”, а для «Denis» = “22”.

Для входу в **Підприємство** використовуються дані з Firebird-таблиці “**USERS**”.

За замовчуванням пароль для користувача «**demo**» = “**22**”.

При першому вході в базу Підприємство поле користувача буде порожнім.

Потрібно додати користувача “**ADD USERS**”.

Користувач має знати своє ім’я й набрати запропонований рядок — у цьому випадку це “**demo**”, далі вводите пароль — і ви в базі!

Якщо ви очистите історію браузера, то знову доведеться додавати користувача.

Тому що дані про користувача зберігатимуться не лише в таблиці “**USERS**”, а й у браузері, щоб кожного разу не бігати в базу!

USERS						
Columns	Constraints	Indices	Triggers	Data	Dependencies	Source
Filter by SQL expression...						Filter by string...
+	ID	ACTUAL	NAME	ROL	PAS	
-	846	<input checked="" type="checkbox"/>	demo	(BLOB)	a73317a3d07ff37fa0837338a29e0c9bd6f488b788be6e98bfc30af9edaf43c	REF (null) SETTINGS (blob)
-	847	<input checked="" type="checkbox"/>	чех1	(BLOB)	a73317a3d07ff37fa0837338a29e0c9bd6f488b788be6e98bfc30af9edaf43c	REF (null) SETTINGS (blob)

Scr3

Для редагування та додавання нових користувачів у Підприємстві є HTML-форма, яка викликається з меню “EditUsers”, як у Конструкторі, так і в Підприємстві.

USER: **demo**

EDIT USERS:

User: **чех1**

847 ACTUAL NAME: **чех1** ROLES: **zakupki**
REF:

PASSW:

Scr4

Управління правами доступу в Підприємстві

- Користувачі з повними правами (“**all**”) можуть змінювати дані, але не можуть змінювати інших користувачів з повними правами.
- Програмісти в Конструкторі — єдині, хто має право змінювати користувачів з повними правами.
- Головний бухгалтер та його заступник можуть отримати повні права, щоб не відволікати програмістів, якщо потрібно додати нового користувача в базу.

Зміна пароля

1. Двічі клацніть у полі “**PASSW**”.
2. Введіть тимчасовий пароль (не довший за п’ять символів).
3. Збережіть зміни, натиснувши “**Save USERS**”.
4. Для введення нового пароля натисніть “**AddNew**” і заповніть усі параметри.
5. Повідомте новий пароль користувачу.
6. Він має увійти в програму з тимчасовим паролем, а потім ввести й підтвердити новий.
7. Новий пароль буде відомий лише користувачу.

Зв’язок з користувачем

У полі **REF** планується зберігати ID довідника типу «Фізичні особи», щоб пов’язати користувача з конкретною людиною.

Початкова база(не путайте з demo)

Початкова база (my4sNull) – це база де ще поки нема метаданих в Підприємстві в таблицях CATS, CATS1, DOCS, REGS. В Конструкторі в таблиці METADATA я вже завів деякі дані, як сподіваюсь, потрібні для правильної роботи особливо для бухгалтерії. А далі ви вже самі повинні знати що робити – це вже ваша програма!

Увага! При першому вході в Конструктор початкової бази пароль — «11». Потім потрібно встановити новий власний пароль, який має бути не менше шести символів.

У Підприємстві на початку роботи у вас не буде жодного користувача (**Users**). Необхідно в Конструкторі спершу ввести себе та свій пароль!

Конструктор.

Після входу в Конструктор з'явиться початкова форма з основним меню ліворуч.

METADATA:

Constructor My4S(ver.4S=1.0.0) for the base mydb(ver.est=1.0.0)			
METADATA:		USER: Программист1	
@ ADD NEW > >		@ Update ↴	
#	NAME	PRESENTATION	ACTUAL
1	BALANSE	(BALANSE) Регістри залишків	✓
2	CATS	CATS (Довідники)	✓
3	CATS1	CATS1 (Підпорядковані Довідники)	✓
4	DOCS	DOCS (Документи)	✓
5	ID_DB	ID_DB: Ідентифікатори бази//	✓
6	Processing	Обробки	✓
7	REGS	Регістри відомостей	✓
8	ROLES	Права(Ролі)	✓
9	Reports	Звіти	✓
10	menu	MENU DB	✓
11	query	Помічник написання звітів	✓

Scr5

Таблиця METADATA та структура меню

Таблиця **METADATA** практично повністю відповідає меню ліворуч, за винятком двох останніх пунктів. Саме на її основі формується меню, яке ви бачите.

Додавання нових об'єктів

Якщо додати новий об'єкт метаданих, він автоматично з'явиться в меню ліворуч.

Однак, щоб він повноцінно працював, потрібно:

Описати його властивості та модулі.

За потреби створити нові таблиці у Firebird.

Для створення програм ERP (аналогічних **1С8**) уже є практично все необхідне.

Якщо знадобиться додаткова функціональність — її можна добавити!

Для початку подивимся ID_DB.

METADATA: ID_DB		USER: Prog1	
@ ADD NEW > >		@ Select ALL ↴	NONE
NAME		PRESENTATION	ACTUAL
D_date	Дата за	//number	✓
accounttype	Вид рах	//text	✓
balans_a	Регістр	//date	✓
balans_b	Регістр	//checkbox	✓
balansreg	Регістр	//select	✓
coef	Коефіці	//balans	✓
dop_param	dop_par	//const	✓
inn	СГРПОУ(ІНН)	//text	✓

Scr6

Основні ідентифікатори метаданих

У програмі вже заздалегідь передбачені основні ідентифікатори (CATS, CATS1, DOCS, REGS та ін.), і краще їх не змінювати.

Під час створення програми програмісти можуть вигадувати власні унікальні імена ідентифікаторів ("Name").

Обмеження на створення ідентифікаторів

- У великих проектах різні програмісти можуть випадково придумати різні ідентифікатори для одних і тих самих елементів, що мають одинакові властивості.
- Щоб уникнути цього, використання нових ідентифікаторів можливе лише через таблицю **ID_DB**.
- Якщо потрібного ідентифікатора там ще немає, його потрібно додати.
- Можна сперечатися про доцільність такого рішення, але я вважаю його найбільш практичним.
- У колонці **PRESENTATION** після опису через // зазначається тип ідентифікатора.
- Перед кожним записом необхідно вибирати тип з випадаючого списку.
- Під час збереження цей тип автоматично додається до рядка в **PRESENTATION**.

Додаткові властивості типів date, select i const

З'явиться нова властивість: **parameter**. Якщо буде потрібно, можна додати нові типи.

Для **date** все очевидно.

У **select**, де:

Необхідно описати випадаючий HTML-список як текст (заміна переліку).

Кожен елемент списку розділений вертикально лінією (|).

Кожен елемент включає два параметри, розділені косою рискою (/):

Перший — сам параметр.

Другий — його представлення.

У **const** необхідно описати значення константи, це може бути число, рядок, рядок JSON, в якому можна описати кілька параметрів. При використанні констант ви повинні знати тип рядка та відповідно використовувати!

Тому в мене немає такого типу метаданих як константи та перерахування!

Edit METADATA:> USER: Программист

METADATA	ID_DB	NAME	vat	PRESENTATION	ПДВ(%)//select	ACTUAL: <input checked="" type="checkbox"/>	select
Parameter:	20/20% 0/0% 7/7% безПДВ						NONE
<input type="button" value="© SAVE >>"/> <input type="button" value="© COPY NEW >>"/>							number
							text
							date
							checkbox
							select
							balans

Scr7

У Підприємстві під час вибору з випадаючого списку це виглядатиме так:

Сума ПДВ	Вся сума	ПДВ(%)	План рахунків
2880	17280	20%	281 Товари на складі
8000	48000	20%	281 Товари на складі
3000	18000	0%	281 Товари на складі
192	1152	7%	281 Товари на складі
14072	84432	безПДВ	

Scr8

BALANSE (Регістри залишків):

Залишки в My4S та їх підтримка у Firebird

- Залишки можуть змінюватися лише документами.
- У Конструкторі передбачені механізми створення рухів регістрів залишків.

У **Firebird** реєстри залишків підтримуються трьома таблицями:

- **BALANSOPT** — зберігає унікальні комбінації вимірів для кожного реєстра залишків, кожна має унікальний ID.

Приклад на номенклатурі:

У всіх документах, які змінюють певну номенклатуру, багаторазово повторюються одні й ті самі комбінації вимірів для кожного реєстра залишків.

- Щоб не зберігати ці комбінації в базі щоразу, їх замінюють унікальними **ID**.
- Це економить місце на диску та пришвидшує роботу системи.

ID	ID_N	W1	W2	W3	W4	W5
795	contr_p_m	806	816			
797	contr_p_m	810	823			
804	account	828	790	777		
805	account	841				
836	corr_account	839				
837	corr_accout	839	843			

Scr9

ID – унікальний ідентифікатор.

ID_N – ім'я реєстра залишків.

W1...W5 – ID елементів довідників (ми вже домовилися, що ID унікальний для будь-якого довідника). Наприклад: **W1** — ID номенклатури, **W2** — ID складу і т.д., як ви налаштуєте в Конструкторі!

BALANSDOC – у цій таблиці зберігаються всі рухи реєстрів залишків за всіма документами.

Для кожного документа може бути будь-яка кількість рядків.

Один рядок — одне проведення, як ви налаштуєте в Конструкторі.

За цією таблицею можна отримати обороти по будь-якому реєстру залишків.

ID_B	ID_DOCS	DAT	ACTUAL	DAT_BALANS	P_M	VV	V1	V2	V3	V4	V5	NUMROW	NUMTAB
797	929	1736642969961	checked	202502	(BLOB)	-85860	0	0	0	0	0	0	0
811	929	1736642969961	checked	202502	(BLOB)	6510	0	0	0	0	0	0	0
812	929	1736642969961	checked	202502	(BLOB)	30000	500	0	0	0	1	1	1
841	929	1736642969961	checked	202502	(BLOB)	30000	0	0	0	0	1	1	1
840	929	1736642969961	checked	202502	(BLOB)	2100	0	0	0	0	1	1	1
832	929	1736642969961	checked	202502	(BLOB)	7350	210	0	0	0	3	1	1
841	929	1736642969961	checked	202502	(BLOB)	7350	0	0	0	0	3	1	1

Scr10

ID_B – зберігає ID унікальної комбінації вимірів з таблиці **BALANSOPT**.

ID_DOCS – ID пов'язаного документа.

DAT – дата документа. У **My4S** майже не використовую вбудований у **Firebird** тип **DATE**, а замість цього застосовую дату як велике число (**BIGINT**), сформоване в **JavaScript**. Усі запити формуються в модулях **JavaScript** — тож навіщо витрачати машинний час на перетворення дати з **JavaScript** у дату **Firebird** і назад!

Хоча в двох таблицях **DOCS** і **REGS** я все ж використовую тип **DATE** у колонці **DATEFB** (без часу). Це потрібно для відборів за періодами дати (день, тиждень, місяць, квартал ...). А у **Firebird** вже є вбудовані функції для роботи з датою!

ACTUAL – ознака, що рух актуальний. Якщо документ під час запису зробити неактуальним, то й усі рухи, пов'язані з цим документом, стануть неактуальними! А залишки будуть сторновані.

DAT_BALANS – усього 6 символів (рік+місяць). Прив'язує рух до конкретного місяця та року.

P_M – вказує, чи це прихід (**true**) або витрата (**false**).

VV – зберігає у текстовому вигляді певні дані, що використовуються надалі, наприклад: “*act=Прихід ТМЦ*”.

V1... V5 – зберігають ресурси, лише тип **NUMERIC**.

Приклад: **V1** — сума руху, **V2** — кількість. Налаштовується в Конструкторі.

NUMROW – номер рядка табличної частини. Якщо **0** — це «шапка» документа.

NUMTAB – унікальний номер табличної частини для всіх рухів по одному документу (може бути до 7 табличних частин).

BALANSE – у таблиці зберігаються залишки на початок кожного місяця, представленого в колонці **DAT_BALANS**.

BALANSE									
Columns		Constraints		Indices		Triggers		Data	Dependencies
									Source
+ Filter by SQL expression...	-	DAT_BALANS	ID_B	P_M	V1	V2	V3	V4	V5
		202503	847	✓	403,36	0	0	0	0 (BLOB)
		202503	823		-256	-3,2	0	0	0 (BLOB)
		202503	824		-3,36	-0,028	0	0	0 (BLOB)
		202504	793		-60225,6	0	0	0	0 (BLOB)
		202504	795		-60225,6	0	0	0	0 (BLOB)

Scr11

Типи колонок уже описані вище, не буду повторюватися.

Початок поточного місяця і кінець попереднього по суті — це один і той самий рядок у таблиці.

Якщо в оборотно-сальдовій задати період початок місяця і кінець місяця, то дані швидко вибирається з рядків цієї таблиці.

А от якщо дата початку чи кінця, або разом, потрапляють між місяцями, тоді для отримання залишків на цю дату будуть використовуватися залишок на початок місяця дати плюс оборот з таблиці **BALANSDOC** від початку місяця до цієї дати.

Важливо для рухів бухгалтерії!

Програма не підтримує зв'язок між рухами дебету і кредиту.

Кожен рух існує самостійно, а за правильність і синхронність рухів відповідає програміст під час опису рухів у документах.

Просто при описі проводок слід описувати попарно: спочатку дебет, а потім кредит (кредит суми з знаком мінус). Це надалі дозволить за допомогою звітів отримати обороти!

Але всі ці проблеми мають вирішувати програмісти, які писатимуть «Бухгалтерію для країни», а **My4S** дозволить зробити майже все, що захочете.

А тепер повернемося до меню й розглянемо роботу в Конструкторі форми **BALANSE (Залишки)**.

METADATA:	BALANSE	USER:	Программист1
<input type="button" value="© ADD NEW METADATA >>"/> <input type="button" value="© Update >"/>			
#	NAME	PRESENTATION	
1	account	БухРахунок	
2	contr_p_m	Рухи грошей по контрагентах	
3	corr_account	Кореспонденційські рахунки	

Scr12

Зверніть особливу увагу на перший рядок: **account** – це зарезервоване ім'я для бухгалтерських залишків. І якщо ви плануєте вести облік залишків за бухрахунками, то цей рядок обов'язково має бути присутнім у такому вигляді, як у демо-версії (PRESENTATION – на ваш розсуд).

Edit Metadata:>	USER:	Программист1	<input type="button" value="© SAVE BALANSE >>"/>	<input type="button" value="© COPY NEW >>"/>			
METADATA:	BALANSE	NAME:	account	PRESENTATION:	БухРахунок	ACTUAL:	<input checked="" type="checkbox"/>
		<input type="button" value="ADD NEW ROW (+)"/> <input type="button" value="DELETE ROW (-)"/>					
#	Type	Name	W_V	PRESENTATION			
1	CATS	accounts	w1	План рахунків		<input type="checkbox"/>	
2	number	sumdoc	v1	Сума документа		<input type="checkbox"/>	

Scr13

Усі інші параметри будуть автоматично братися з довідника бухрахунків (План рахунків) accounts (також зарезервований ідентифікатор!).

Хочу звернути особливу увагу на те, що в My4S немає окремо реєстру бухгалтерії та інших реєстрів залишків. Усе працює як єдина система з трьох таблиць, як описано вище. Просто бухзалишки відрізняються лише тим, що перший вимір (W1) завжди є посиланням на довідник бухрахунків (План рахунків). І в поєднанні з зарезервованими account та accounts сам код уже розбирається, як працювати з бухгалтерськими підсумками. Далі ви можете створювати будь-яку кількість простих реєстрів залишків, присвоюючи їм вигадані вами ідентифікатори, описуючи всі виміри та ресурси на власне розуміння створюваної вами програми.

Edit Metadata:> USER: Программист1

SAVE BALANSE >> COPY NEW >>

METADATA: BALANSE NAME: contr_p_m PRESENTATION: Рухи грошей по контрагентах ACTUAL:

ADD NEW ROW (+) DELETE ROW (-)

#	Type	Name	W_V	PRESENTATION	
1	CATS	contr	w1	▼ Контрагенти	<input type="checkbox"/>
2	CATS1	contract	w2	▼ Договір контрагента	<input type="checkbox"/>
3	number	sumdoc	v1	▼ Сума документа	<input type="checkbox"/>

Scr14

Edit Metadata:> USER: Программист1

SAVE BALANSE >> COPY NEW >>

METADATA: BALANSE NAME: corr_account PRESENTATION: Кореспонденційські рахунки ACTUAL:

ADD NEW ROW (+) DELETE ROW (-)

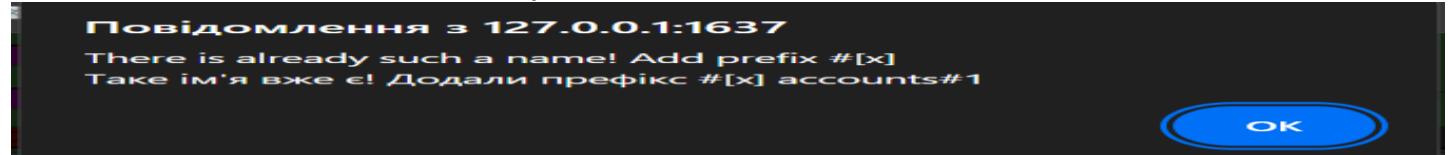
#	Type	Name	W_V	PRESENTATION	
1	CATS	accounts	w1	▼ План рахунків дебіт	<input type="checkbox"/>
2	CATS	accounts#1	w2	▼ План рахунків кредит	<input type="checkbox"/>
3	number	sum1	v1	▼ Сума	<input type="checkbox"/>

Scr15

Чому accounts і accounts#1.

У вас виникне питання, чому accounts і accounts#1. Це ще одна моя «фішка». Коли в Конструкторі ви обираєте якийсь Type, то вам випаде вже готовий список ідентифікаторів під цей тип, і там ви ніколи не побачите ідентифікатора з закінченням (#n). Річ у тім, що цю приставку (#n) створює сама програма. Перед створенням нового рядка програма перевіряє Name усіх рядків на збіг з новим ідентифікатором без урахування закінчення (#n), і при цьому підсумовує всі збіги. Якщо потрібно, програма сама допише нове закінчення до ідентифікатора.

Перед цим видавши повідомлення:



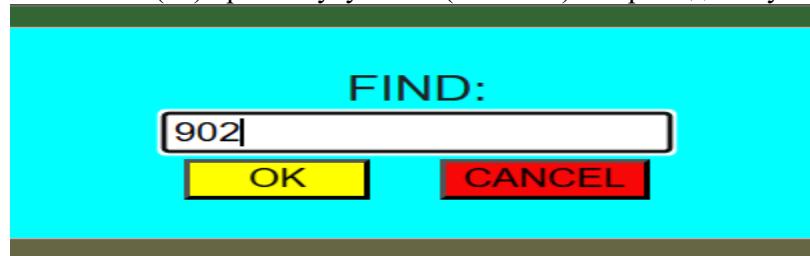
Scr16

Ця «фішка» потрібна для пошуку в Підприємстві різних елементів довідників (CATS) і не тільки. У кожній комірці таблиці HTML є текстовий параметр «title». Якщо йому присвоїти якесь значення, то при наведенні курсора на цю комірку під нею з'явиться текст «title». Саме цей параметр буде заповнюватися програмою в момент заповнення таблиці «на льоту», спираючись на дані, записані в конструкторі. Цей параметр буде записуватися в комірки, де є можливість вибору з любого довідника. Рядок матиме вигляд: METADATA|id_n. На наступному зображені «title» = «CATS|accounts#1».

ПДВ(%)	План рахунків
20%	902 Собівартість реалізованих
5	CATS accounts#1

Scr17

І тепер під час пошуку елемента у довіднику програма знає, у якому саме довіднику шукати цей елемент. Для цього програма відкидає закінчення (#1) при пошуку в базі (Select ...) та при заданому вами відборі (WHERE).



Scr18

Повертається список відібраних елементів (не більше 50!), з яких ви оберете потрібний, а якщо елемент виявиться один, то він одразу ж потрапить у комірку вибору! У рядку пошуку можна писати кілька параметрів, розділених пробілом. Параметри – це рядки, відбираються ті елементи, в яких будуть знайдені всі входження з рядка пошуку. Пошук здійснюється за найменуванням елемента або за його кодом. Є особливість: якщо перший символ є цифрою, то пошук відбувається за параметром "cod", а інакше – за "name"!

Приклад: «Пиво Темне 0.5» – у рядку пошуку можна написати «пив тем 5» і ви отримаєте саме те, що потрібно. Для пошуку за штрихкодом відскануйте його у рядок пошуку, тоді пошук буде здійснюватися за параметром "cod". Приблизно такий вигляд може мати рядок відбору при відборі з Номенклатури:



Scr19

Робота форми з налаштуванням метаданих.

На прикладі довідників (CATS) розглянемо, як працювати з формами.

METADATA:		USER:	
CATS		Программист1	
<input type="button" value="© ADD NEW >>"/>		<input type="button" value="© Update ⌂"/>	
#	NAME	PRESENTATION	ACTUAL
1	contr	Контрагенти	✓
2	nomencl	Номенклатура	✓
3	nomenclgroup	Номенклатурні групи	✓
4	accounts	План рахунків	✓
5	store	Склади(Місця зберігання)	✓

Scr20

Для початку редагування будь-якого довідника – зробіть подвійне класання на ньому. Покажемо на номенклатурі:

The screenshot shows the 'Edit Metadata' interface. At the top, there are fields for 'USER' (Programmist1), 'ROLES' (buh,zakupki), and buttons for 'SAVE METADATA >>' and 'COPY NEW >>'. Below this, a 'METADATA' section shows 'NAME: nomencl' and 'PRESENTATION: Номенклатура'. A 'Select Form:' dropdown is set to 'TITLE'. The main area is a table with columns: #, In journal, Type, Name, Width %, Title, Read only, Hidden, ADD Attributes, Layers, and a checkmark column. The table contains five rows of data:

#	In journal	Type	Name	Width %	Title	Read only	Hidden	ADD Attributes	Layers	
1	<input type="checkbox"/>	select	vat	5	ПДВ(%)	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
2	<input checked="" type="checkbox"/>	select	typesnomenc	8	Види номенклатури	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
3	<input type="checkbox"/>	select	unit	8	Одиниця виміру	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
4	<input type="checkbox"/>	text	name1	12	ІншеНайменування	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
5	<input checked="" type="checkbox"/>	CATS	accounts	8	План рахунків	<input type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>

Scr21

Усе, що відображене рожевим кольором, для редагування закрите! Тому перед створенням будь-яких нових об'єктів добре подумайте!

Одразу ж розповім про кнопку “COPY NEW”. При натисканні ця форма буде скопійована й створена як нова форма ще не записаного в базу метаданого. У полі «NAME» буде рядок-підказка “NEW METADATA!?”’. Замість нього ви впишете ім’я, придумане вами для метаданого.

Усе рожеве — в силі. Але можна видалити рядок і створити новий, над новою формою можна експериментувати без наслідків, доки не запишете (SAVE). Для видалення рядків є кнопка “DELETE ROW()”. Сама кнопка під час натискання нічого не видалить. Треба її підказати, які рядки видаляти. Для цього є остання колонка — «галочка», якою ви вказуєте, який рядок видаляти, або кілька рядків, і після цього можете видаляти. галочка також потрібна для зсуву рядків під час натискання кнопок «вниз» і «вгору», але якщо поставимо дві галочки і більше — зсув не працюватиме! **Зсув не працює по колу!** Так працює у всіх формах Конструктора та Підприємства, де є «галочка» і кнопка “COPY NEW”!

Розглянемо Select Form: у ньому є радіо-кнопки TITLE, CODE, DESCRIPTION. Коли я починав, то думав використовувати їх усі, але поки мені знадобилася лише одна – TITLE. Решту можна використовувати для збереження певної текстової інформації у blob- полях WW (CODE) та DESCRIPTION у таблиці METADATA.

TITLE, як ви могли помітити, це HTML-таблиця (table). Кожен рядок описує одну колонку в табличній частині таблиці, як вона виглядатиме у Підприємстві, коли відкриєте довідник номенклатура. Скільки рядків у таблиці – стільки колонок у Підприємстві. У моїй демо-версії це виглядатиме так:

The screenshot shows the 'Edit Element' interface. At the top, there are fields for 'METADATA(NAME): Номенклатура', 'USER: demo', 'ID: 798', 'GROUP: (780)GCM', and buttons for 'SAVE ELEMENT >>' and 'COPY NEW >>'. Below this, a 'ELEMENT(name): Бензин-92' field and a 'COD: 92' field are shown. A 'INFO:' field is empty. The main area is a table with columns: ПДВ(%), Види номенклатури, Одиниця виміру, ІншеНайменування, and План рахунків. The table contains one row of data:

ПДВ(%)	Види номенклатури	Одиниця виміру	ІншеНайменування	План рахунків
7%	ГСМ	Кілограми	Бензин-92	203 Паливо

Scr22

Як я вже раніше описував, це ви бачите HTML-шаблон (ELEMENT.htm), верхня частина підіде для будь-якого довідника, а от нижню таблицю ми опишемо в Конструкторі, як вона виглядатиме для конкретного довідника. У даному випадку – для номенклатури.

Повернемося знову до Конструктора. Розглянемо кожну колонку:

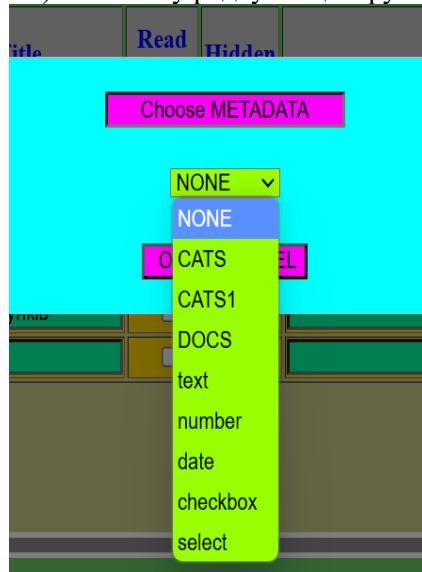
1. Номер за порядком
2. In journal – вказує, чи виводити інформацію про рядок у журнал, у Підприємстві це виглядатиме так:

The screenshot shows the 'Edit Group' interface. At the top, there are fields for 'METADATA(NAME): Номенклатура', 'USER: demo', 'ID: 780', 'GR: 0', and 'ACTUAL: '. Below this, a 'GROUP(name): GCM' field and a 'SAVE GROUP >>' button are shown. The main area is a table with columns: #, ELEMENT (name), COD, ACTUAL, and a combined 'Види номенклатури|План рахунків' column. The table contains four rows of data:

#	ELEMENT (name)	COD	ACTUAL	Види номенклатури План рахунків
1	Бензин-92	92	<input checked="" type="checkbox"/>	GCM 203 Паливо
2	Бензин-95	95	<input checked="" type="checkbox"/>	GCM 203 Паливо
3	Бензин-98	98	<input checked="" type="checkbox"/>	GCM 203 Паливо
4	Оліва М10Г2К	10Г	<input checked="" type="checkbox"/>	GCM 203 Паливо

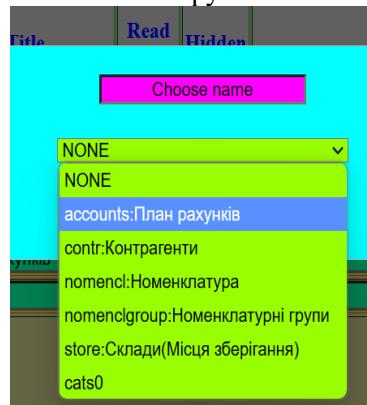
Scr23

3. Type – можна змінювати лише під час додавання нового рядка ADD NEW ROW (). При подвійному кладанні (ondblclick) на новому рядку по центру з'явиться меню:



Scr24

4. Name – залежно від вибору з першого меню з'явиться ще одне меню. Припустимо, ви обрали CATS, тоді випадатиме список усіх довідників: cats0 – це перенесений вибір довідника вже у Підприємстві перед початком вибору значення.



Scr25

Якщо обрано CATS1 – підпорядкований довідник. Він обов'язково йде в парі з основним довідником CATS. Спочатку йде CATS і одразу за ним CATS1, інакше працювати не буде (Контрагенти, договори)! Якщо обрано DOCS, тоді випадатиме список усіх документів. А все, що йде після DOCS: text і далі за списком – це імена ідентифікаторів, відібраних за типом з таблиці ID_DB, яка описувалася вище. Width % - задається ширина ячейки.

5. Title – текст заголовка колонки таблиці, якщо не поміщається, буде переноситися в комірці, розширяючи її по вертикалі.

6. Read only – заборона на редагування вмісту комірки.

7. Hidden – чи буде видимою комірка.

8. ADD Attributes – можна додатково описати будь-який атрибут, окрім наведених вище, за правилами HTML! Уся потужність HTML у ваших руках і знаннях. Але мені, як бачите, поки не знадобилося використовувати можливості цієї колонки. Досить багато того, що ви зможете побачити в демо-версії, записано у файлі TabFix.css, який ви також зможете вдосконалити на свій смак і колір!

9. Layers (Шари) – у мене всі форми таблиць за замовчуванням розтягаються на всю ширину екрана. І якщо колонок мало, то цей пункт не потрібен. А коли їх багато, то браузер почне автоматично пропорційно виставлені вами ширині комірки (Width) підганяти, щоб усі колонки вписалися у ширину екрана. У колонку Layers записуємо ім'я шару (просто рядок будь-якою мовою). Якщо ми хочемо, щоб деякі колонки завжди були видимі в кожному шарі, просто залишаємо Layers порожнім. А решту колонок ділимо на шари, вказуючи кожному шару своє унікальне ім'я (шарів може бути кілька). При виведенні форми у Підприємстві над таблицею ліворуч з'явиться випадаючий список усіх шарів.

10. «галочка» – я вище розповідав, для чого вона потрібна.

Комплекти.

У My4S підтримується такий підвід довідника, як «Комплекти». Ознакою комплекту в Підприємстві є відкриваюча та закриваюча дужки «()» наприкінці найменування елемента довідника. Обов'язково в Конструкторі має бути створений підпорядкований довідник (CATS1) та описані колонки.

Edit Metadata >	USER: Программист 1	ROLES: buh,zakupki	<input type="button" value="© SAVE METADATA >>"/>	<input type="button" value="© COPY NEW >>"/>							
METADATA: CATS1	NAME: kit	PRESENTATION: Комплекти	OWNER: nomenc:Номенклатура	ACTUAL: <input checked="" type="checkbox"/>							
Select Form:											
<input type="radio"/> TITLE <input type="radio"/> CODE <input type="radio"/> DESCRIPTION											
TITLE: <input type="button" value="ADD NEW ROW (+)"/> <input type="button" value="DELETE ROW (-)"/>											
#	In journal	Type	Name	Width %	Title	Read only	Hidden		ADD Attributes	Layers	<input checked="" type="checkbox"/>
1	<input checked="" type="checkbox"/>	CATS	nomenc	50	Номенклатура	<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
2	<input checked="" type="checkbox"/>	number	kount	12	Кількість	<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>

Scr26

А в Підприємстві:

<< EDIT GROUP(CATS) >>					<input type="button" value="© SAVE GROUP >>"/>
METADATA(NAME): Номенклатура		•USER: demo•	•ID: 781•	•GR: 0 •	ACTUAL: <input checked="" type="checkbox"/>
GROUP(name): КОМПЛЕКТИ					
<input type="button" value="© ADD NEW ELEMENT >>"/> <input type="button" value="Move to group >>"/> <input type="button" value="© Update >>"/>					
#	ELEMENT (name)	COD	ACTUAL	<input checked="" type="checkbox"/>	Види номенклатури План рахунків
1	Капучино()	4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Комплект(2821 Товари в роздрібній торгівлі (в АТТ за продажною вартістю))
2	Мокаччино()	44	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Комплект(2821 Товари в роздрібній торгівлі (в АТТ за продажною вартістю))

Scr27

<< EDIT ELEMENT(CATS) >>					<input type="button" value="© SAVE ELEMENT >>"/>	<input type="button" value="© COPY NEW >>"/>
METADATA(NAME): Номенклатура		•USER: demo•	•ID: 799•	•GROUP: (781)КОМПЛЕКТИ•	COD: 4	ACTUAL: <input checked="" type="checkbox"/>
ELEMENT(name): Капучино()						
INFO:						
Values (VV):						
ПДВ(%)	Види номенклатури	Одиниця виміру	ІншеНайменування	План рахунків		
20%	Комплект	штуки	Капучино_ Комплект	2821 Товари в роздрібній торгівлі (<input type="button" value="© OPEN Subordinate >>"/>	

Scr27a

І відкриваємо підпорядкований довідник «OPEN Subordinate»:

<< EDIT GROUP subordinate(CATS1) >>					<input type="button" value="© SAVE ELEMENT >>"/>
USER: demo•		•GROUP: (799)Капучино•			
<input type="button" value="© ADD NEW ELEMENT >>"/> Комплекти					
#	ID_Name	ELEMENT (name)	ACTUAL	Номенклатура кількість	
1	kit	Замінник молока#	<input checked="" type="checkbox"/>	Замінник молока 0.8	
2	kit	КофеTotti Supremo#	<input checked="" type="checkbox"/>	КофеTotti Supremo 0.007	

Scr28

А потім редагуємо або створюємо новий елемент:

<< EDIT SUBORDINATE ELEMENT(CATS1) >>					<input type="button" value="© SAVE ELEMENT >>"/>	
METADATA(NAME): Комплекти		•USER: demo•	•ID: 821•	•ONWER: (799)Капучино•		
ELEMENT(name): Замінник молока#					ACTUAL: <input checked="" type="checkbox"/>	
INFO:						
Values (VV):						
Номенклатура	Кількість					
Замінник молока	0.8					

Scr29

І наприкінці імені кожного елемента комплекту підпорядкованого довідника обов'язково ставимо символ “#”.

Уточнимо - для всіх елементів ROLES у Конструкторі – якщо він порожній, це означає повний доступ усіх користувачів у Підприємстві до цієї форми! Нагадаю, рожевий колір не можна змінювати (Read only). Але є виняток для деяких елементів форми. Розглянемо елемент ROLES. Формально він дійсно є Read only і тому має рожевий колір. Але нам усе ж потрібно мати можливість накласти на об'єкт права користувачів, які мають можливість його змінювати (не забуваймо, користувач з правом «all» має повний доступ до всіх об'єктів метаданих без обмежень і немає потреби вказувати це право зайвий раз!).

Для накладання прав на об'єкт клацніть на елементі ROLES. Відкриється список усіх прав, які були введені в меню «Права (Ролі)»:

Scr30

Якщо потрібно відкоригувати або видалити список уже вибраних прав, клацніть на Empty, а потім клацніть поза цією формочкою – список буде видалено! Потім знову клацніть на елементі ROLES і, утримуючи клавішу Shift, можна вибрати кілька прав одночасно, а після завершення вибору клацніть поза цією формочкою – у полі ROLES з'явиться список вибраних прав, розділених комами. Майже все описане для довідників (CATS) щодо роботи з формами буде працювати й в інших формах у Конструкторі.

Підпорядкований довідник (CATS1).

Усе те саме, що й для CATS, тільки є одна особливість – це OWNER, ім'я довідника, якому він підпорядкований.

Scr31

Регістри відомостей (REGS).

Усе те саме, що й для CATS. Поки що REGS відрізняються від довідника лише тим, що в Підприємстві під час створення елемента є можливість ставити дату, і це автоматично робить регістри відомостей періодичними!

У Конструкторі:

Scr32

В Підприємстві:

Scr33

В firebird:

Scr34

регистри відомостей – це окремий об'єкт для зберігання даних. На нього неможливо зробити посилання в метаданих. Як ви могли помітити, у виборі Type немає такого типу! Отримати дані можливо з firebird, лише виконавши (Select ...)!

Документи (DOCS).

Це найскладніший з об'єктів метаданих. Окрім шапки він має ще й табличні частини (може мати до 7). У документах описуються всі рухи регистрів. Можна призначити, на підставі яких документів вводиться даний документ (IS THE BASIS FOR).

#	In journal	Type	Name	Width %	Title	Read only	Hidden	ADD Attributes	Layers	
1	<input checked="" type="checkbox"/>	CATS	contr	12	Контрагент	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input checked="" type="checkbox"/>
2	<input type="checkbox"/>	CATS1	contract	12	Договір контрагента	<input type="checkbox"/>	<input checked="" type="checkbox"/>			<input type="checkbox"/>
3	<input checked="" type="checkbox"/>	CATS	store	12	Склад	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
4	<input type="checkbox"/>	CATS	accounts	12	План рахунків	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
5	<input type="checkbox"/>	number	sumvatdoc	7	Сума ПДВ документа	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
6	<input checked="" type="checkbox"/>	number	sumdoc	8	Сума документа	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
7	<input type="checkbox"/>	checkbox	prices_vat	4	Ціни з ПДВ	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>

IS THE BASIS FOR: NONE

Scr35

Початкові заповнення таблиць, як у шапці, так і в табличних частинах, повністю відповідають описаному вище в CATS. Є лише невелика відмінність: під час заповнення табличних частин (TABLES) замість колонки «In journal» з'явиться колонка «tfoot», яка вказуватиме, які колонки в Підприємстві потрібно сумувати та виводити цю суму в нижню підсумкову колонку.

#	tfoot	Type	Name	Width %	Title	Read only	Hidden	ADD Attributes	Layers	
1	<input type="checkbox"/>	CATS	nomencl	12	Номенклатура	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
2	<input type="checkbox"/>	number	kount	6	Кількість	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
3	<input type="checkbox"/>	number	coef	4	Коефіцієнт	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
4	<input type="checkbox"/>	text	price	5	Ціна	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
5	<input checked="" type="checkbox"/>	number	sumcost	7	Сума без ПДВ	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
6	<input checked="" type="checkbox"/>	number	sumvat	6	Сума ПДВ	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
7	<input checked="" type="checkbox"/>	number	sumrow	8	Вся сума	<input checked="" type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
8	<input type="checkbox"/>	select	vat	5	ПДВ(%)	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
9	<input type="checkbox"/>	CATS	accounts#1	12	План рахунків	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>

IS THE BASIS FOR: NONE

Scr36

В Підприємстві буде виглядати так:

Контрагент	Договір контрагента	Склад	План рахунків	Сума ПДВ документа	Сума документа	Ціна з ПДВ
Топікін Микола Михайлович ПІ	Договір №124 від 24.03.2022	Головний склад	631 Розрахунки з вітчизняними	14072,0000	84432,0000	

TABLES : Товари

#	Номенклатура	Кількість	Коефіцієнт	Ціна	Сума без ПДВ	Сума ПДВ	Вся сума	ПДВ(%)	План рахунків
1	Кофетotti Supremo	120	1	120	14400	2880	17280	20%	281 Товари на складі
2	Замінник молока	500	1	80	40000	8000	48000	20%	281 Товари на складі
3	Горічкий Шоколад	150	1	100	15000	3000	18000	20%	281 Товари на складі
4	Коняк "Наполеон"	4	1	240	960	192	1152	20%	281 Товари на складі

Scr37

Повернемося до Конструктора:

Debet_Credit: визначає документ за основним призначенням — прибутковий (Debet) чи видатковий (Credit). Якщо встановлено галочку — це прибуток, інакше витрата.

Select Form: перемикач між частинами документа, з якими працюємо в даний момент.

Кожен документ має шапку (TITLE) та багаторядкові частини (TABLES). Шапка — це обов'язковий елемент, а багаторядкових таблиць може й не бути!

MaketPrint: просте поле `textarea`. При створенні нового документа воно порожнє. Його рекомендується заповнювати наприкінці редагування документа, а краще вже після перевірки роботи документа у Підприємстві. При відкритті цієї форми ви побачите дві кнопки — “Generate print” і “View print”. Перша кнопка при натисканні згенерує друковану форму документа у текстовому вигляді на чистому HTML-коді. Друга кнопка покаже, як виглядатиме ця форма у браузері (лише структура). У нашому випадку дві табличні частини — отже буде показано дві заготовки.

..txt..										
Контрагент: ..contr.. Договір контрагента: ..contract.. Склад: ..store.. План рахунків: ..accounts..										
#	Номенклатура	Кількість	Коефіцієнт	Ціна	Сума без ПДВ	Сума ПДВ	Вся сума	ПДВ(%)	План рахунків	
..#..	..nomencl..	..kount..	..coef..	..price..	..sumcost..	..sumvat..	..sumrow..	..vat..	..accounts#1..	
Сума без ПДВ: ..5.. Сума ПДВ: ..6.. Вся сума: ..7..										

//TAB

..txt..										
Контрагент: ..contr.. Договір контрагента: ..contract.. Склад: ..store.. План рахунків: ..accounts..										
#	Номенклатура	Кількість	Ціна	Сума без ПДВ	Сума ПДВ	Вся сума	ПДВ(%)	План рахунків		
..#..	..nomencl..	..kount..	..price..	..sumcost..	..sumvat..	..sumrow..	..vat..	..accounts#1..		
Сума без ПДВ: ..4.. Сума ПДВ: ..5.. Вся сума: ..6..										

//TAB

Scr38

А в Підприємстві, спираючись на цю заготовку, буде формуватися друкована форма, вже заповнена на підставі даних документа. Якщо в документі кілька табличних частин, то друкована форма формуватиметься лише за тією таблицею, вкладку якої буде вибрано на даний момент!

Прибуткова накладна #000000000004 > 2025-03-02 15:03:04.241(Товари)

Контрагент: БестТрейд Договір контрагента: Договір №250 від 20.10.2020 Склад: Головний склад План рахунків: 631 Розрахунки з вітчизняними постачальниками										
#	Номенклатура	Кількість	Коефіцієнт	Ціна	Сума без ПДВ	Сума ПДВ	Вся сума	ПДВ(%)	План рахунків	
1	Коньяк "Наполеон"	120	1	240	28800	5760	34560	20%	281 Товари на складі	
2	Горілка Козацька рада "Оригінальна" 0,75 л	210	1	100	21000	4200	25200	20%	281 Товари на складі	
Сума без ПДВ: ..49800.00 Сума ПДВ: ..9960.00 Вся сума: ..59760.00										

Scr39

Редагування текстового поля (textarea)

Ви можете вільно редагувати текст у полі `textarea` на власний розсуд. Під час збереження документа в Конструкторі ваші зміни будуть зафіксовані. Якщо результат редагування вас не влаштовує, можна скасувати зміни, видаливши весь текст з поля (`Ctrl+A → Delete`), після чого натисніть кнопку “Generate print” — автоматично буде сформовано новий (початковий) текст.

Щоб повернутися до редагування документа, просто клацніть по його заголовку. У Конструкторі це буде «..txtt..», у системі Підприємство — «Прибуткова накладна ...».

Колонка «ADD Attributes»

Навпроти параметра «Ціна» вказано:

`value=0 style="text-align: center;"`

- **value** — значення, яке буде встановлено за замовчуванням при відкритті форми.
- **style="text-align: center;"** — визначає вирівнювання значення по центру комірки. Це частина стандарту HTML, а не авторська розробка.

Атрибут `data-r="vat, accounts"`

Це моя доробка — **data-r** є зарезервованим ідентифікатором. У Підприємстві при виборі номенклатури в рядку програма перевіряє, чи є такий ідентифікатор (**data-r**). Якщо він є, то програма аналізує, чи має ця номенклатура ідентифікатори (**vat, accounts**). Якщо так, програма запам'ятовує ці значення. І якщо

знаходить такі самі ідентифікатори в рядку документа, вони автоматично заповнюються! Потім їх, звісно, можна змінити. Якщо одразу не зрозуміло — у демоверсії стане зрозуміло.

Вкладка «DESCRIPTION»

Це просто поле **textarea**, куди можна ввести довільний текст: Help, коментарі, підказки та інше. Ніде, окрім цієї вкладки, цей текст не використовується.

Нове (Excel)!

З'явилася нова можливість підготовки шаблонів (Макетів) за допомогою Excel.xlsx. Звичайно, макет можна було підготувати й за допомогою HTML (за замовчуванням у My4s простий звіт за табличними частинами створюється майже автоматично, але не завжди цього вистачає). Але витрати часу на підготовку такого макета у багато разів менші, ніж на підготовку такого звіту в Excel (наприклад, такі макети як Податковий рахунок-фактура, Товарно-транспортні накладні тощо)!

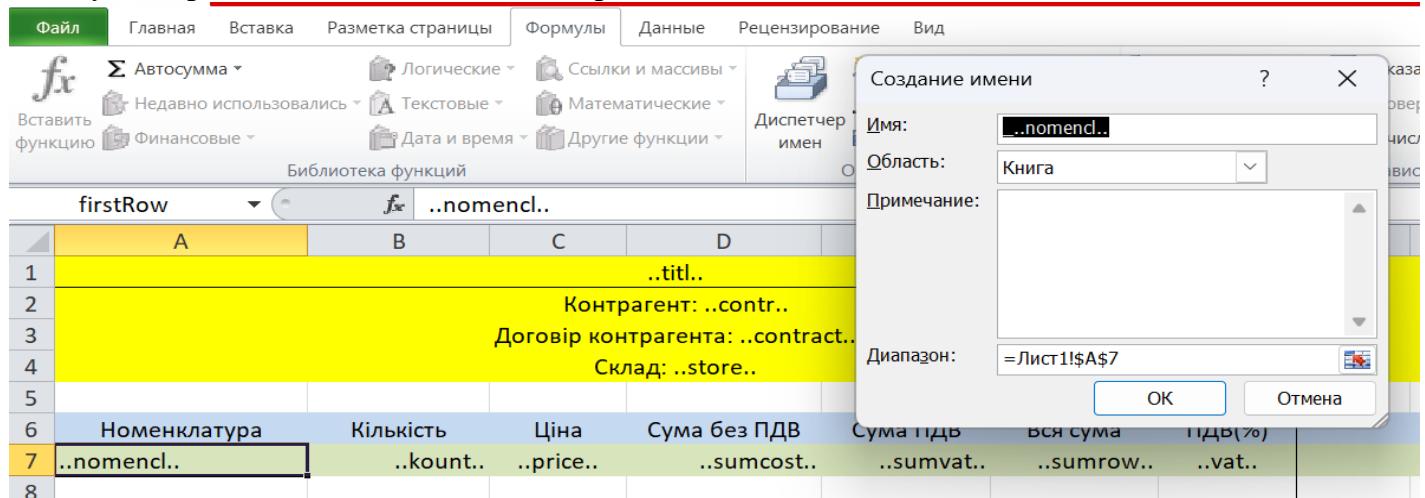
Тепер розберемося, як це працює поки що у Конструкторі. Спочатку має бути встановлено Excel.xlsx. Спочатку створюємо макет, як він має виглядати на друк.

Якщо має бути присутня таблична частина, створюємо шапку і один рядок (уже в Підприємстві; якщо в табличній частині буде кілька рядків, вони автоматично запищуться як треба, якщо правильно підготовлений макет). У макеті, ймовірно, буде багато клітинок, які будуть заповнюватися даними з документа під час формування друкованої форми. У ці клітинки треба вписати імена ідентифікаторів даних з документа в Конструкторі та обмежити їх подвійними крапками.

Приклад: ..contr.. (Назва контрагента з шапки “TITLE”). У табличній частині ..nomencl.. (Назва номенклатури з “TABLES”).

Стилі клітинок ви можете налаштовувати самі в макеті (колір, шрифт, вирівнювання ліво-право, число/текст, перенесення, якщо не поміщається в клітинку і багато іншого).

І ще дуже важлива річ. У шаблоні Excel потрібно вказати номер рядка, з якої починається таблична частина (тільки не шапка!). Для цього ставите курсор на першу колонку табличної частини; у нашому випадку це ..поменcl.., натискаємо **Ctrl+F3** — з'явиться вікно **Диспетчера імен**, у ньому натискаємо кнопку **Створити** — з'явиться вікно **Створення імені**:



Scr39a

Ім'я обов'язково створюємо **firstRow**. І дуже важливо вибрати діапазон (там уже має стояти правильний діапазон, якщо ви поставили курсор туди, куди потрібно) або вибрати заново, натиснувши кнопку з правого боку на рядку Діапазон. Такий вибір діапазону дозволяє у подальшому, якщо ви буде допрацьовувати шаблон і додасте над табличною частиною ще одну нову строку, діапазон автоматично зсуватиметься на потрібну кількість рядків! Як ви вже побачили на знімку, у клітинках можна писати що завгодно, але підставлятися буде лише частина, заключена у подвійні крапки! Якщо все правильно заповните, то в Підприємстві програма самостійно сформує потрібну вам форму для друку. І це ще не все! Потрібно вказати програмі, що у нас є шаблони для Excel. Для цього переходимо в MaketPrint і в кінці дописуємо рядок виду: //excel Print EXCEL1//excel Якщо у вас повинно бути два шаблона

документа:

//excel Print_EXCEL1//excel Print_EXCEL2//excel Якщо більше двох — здогадаєтесь самі!
Замініть Print_EXCEL1 на будь-яке ім'я, яке ви придумаєте - воно з'явиться в меню друку документа в Підприємстві:

<< EDIT DOCUMENT >>

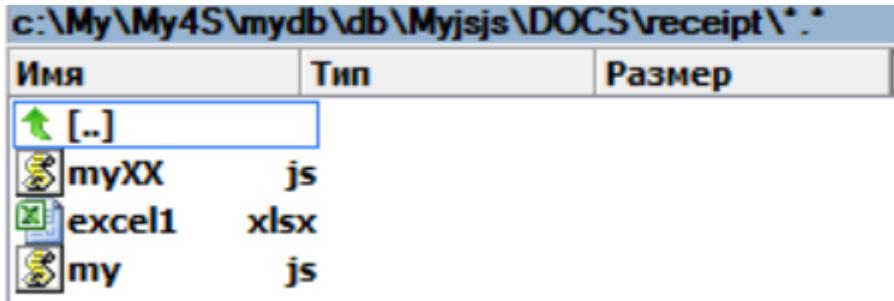
Прибуткова накладна*	• USER: demo*	• ID: 929*	PRINT >>					
NumDoc: 000000000003	DateDOC: 07.01.2026	T 00:49:29.961	ACTUAL: <input checked="" type="checkbox"/>	PRINT >>				
INFO:		BALANSE Document						
Контрагент		Договір контрагента		Склад				
Рабко Сергій Георгіевич		Дог. 22 від 19.03.2023		Головний склад				
				631 Розр.				
				Print_EXCEL1				
				Print_EXCEL2				
Сума ПДВ документ								
6610								
TABLES : Товари								
ADD ROW (Alt+)		DELETE ROW (Alt-)						
#	Номенклатура	Кількість	Коефіцієнт	Ціна	Сума без ПДВ	Сума ПДВ	Вся сума	ПДВ(%)
1	Бензин-92	500	1	60	30000	2100	32100	7%
2	Бензин-95	600	1	70	42000	2940	44940	7%
3	Оліва М10Г2К	210	1	35	7350	1470	8820	20%
					79350	6510	85860	

Scr39b

Шаблони Excel завжди мають називатися excel1.xlsx, excel2.xlsx тощо. Вони відрізнятимуться лише кількістю в порядку меню:

//excel Print_EXCEL1//excel Print_EXCEL2//excel

Тепер розберемося, куди розмістити шаблон або шаблони Excel. І ці файли потрібно скопіювати до папки з метаданими документа, у нашому випадку це папка receipt, в якій щонайменше має бути вже файл myXX.js.



Scr39c

Детально про ці папки у мене описано далі в розділі **Внутрішні папки бази**.
Тому не буду повторюватися!

Так як у нас My4S являється web-застосунком, то браузер поки що не може читати файли ексел. Тому програма на льоту зберігає файл як PDF-файл з тим же іменем але з розширенням .pdf, А браузер вже вміє читати файли PDF.

Опис рухів registrів

Цей етап — ключовий у роботі з документами. До опису рухів слід приступати після формування документа в Конструкторі.

У шапці та табличних частинах документа є поле **BALANSE** з випадаючим списком.

Поруч знаходиться текстове поле, у яке автоматично записуються рядки, що описують рухи регистрів (розділені символом ; ;).

Це поле візуально виділене кольором і не призначено для прямого редагування. Однак при подвійному класанні по ньому відкриється додаткове поле **textarea**, де всі рухи будуть представлені построчно.

Це поле можна редагувати вручну — за умови дотримання правил запису рухів.

TITLE TABLES MaketPrint DESCRIPTION

1 2 ADD TABLE >

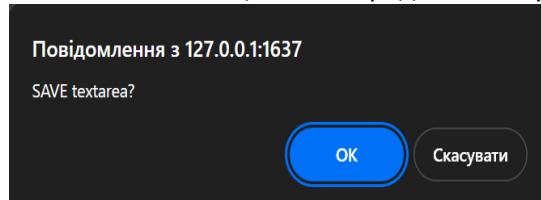
BALANSE: NONE

```

account/true/w1=accounts#1/w2=nomencl/w3=store/v1=sumcost/v2=kount/vv=act=ТМЦ+;;account/false/w1=accounts/w2=contr/w3=contract/v1=sumcost/vv=act=КОНТР-;;
account/true/w1!=6432!/w2=contr/w3=contract/v1=sumvat/vv=act=ПДВ+;;
account/false/w1=accounts/w2=contr/w3=contract/v1=sumvat/vv=act=КОНТР-;;
corr_account/true/w1=accounts#1/w2=accounts/v1=sumcost;;
corr_account/true/w1!=6432!/w2=accounts/v1=sumvat;;
    
```

Scr40

Для збереження відредагованого поля клацніть на нередагованому полі — з'явиться форма:



Scr41

Якщо натиснути **ОК**, відредаговані дані з'являться в нередагованому полі як один рядок, розділений символом «;;».

А тепер подивимось, як це працює:

Select Form:

TITLE TABLES MaketPrint DESCRIPTION

BALANSE: NONE

		ADD NEW ROW ()		DELETE ROW ()	
#	In journal	Width %	Title	Read only	Hidden
1	<input checked="" type="checkbox"/>	12	Контрагент	<input type="checkbox"/>	<input type="checkbox"/>
Empty account:БухРахунки contr_p_m:Рухи грошей по контрагентах corr_account:Кореспонденційські рахунки					

Scr42

Меню BALANSE

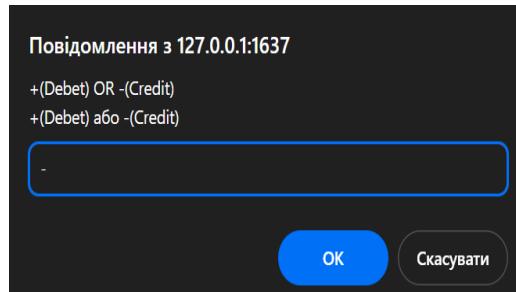
У випадаючому списку **BALANSE** представлено меню всіх дій при формуванні рухів.

Якщо натиснути **Empty**, то нередагований рядок буде очищено, і його вже так просто не повернеш — доведеться з нуля заповнювати рухи регістрів!

Далі буде наведено повний список регістрів залишків, як вони описані в **Метадата BALANSE**.

Початок роботи з регістрами залишків

Припустимо, вибрали **contr_p_m**. З'явиться вікно для введення значень:



Scr43

Введення символу для руху

Тут потрібно вказати один символ: + (Прихід) або – (Витрата).
Якщо натиснути будь-який інший символ — це буде помилка!

Далі слід натиснути **Enter** або **OK**, а якщо хочете скасувати дію — натисніть **Esc** або **Скасувати**.

Якщо все виконано правильно, то в нередагованому полі з'явиться новий рядок:
(contr_p_m/false/w1=contr/w2=contract/v1=sumdoc/; ;)

Цей рядок автоматично генерується програмою на основі даних у **Метадата BALANSE** для вибраного реєстра.

Edit Metadata:>				USER: Программист1	SAVE BALANSE >>	COPY NEW >>
METADATA: BALANSE		NAME: contr_p_m	PRESENTATION: Рухи грошей по контрагентах	ACTUAL: <input checked="" type="checkbox"/>		
#	Type	Name	W_V	PRESENTATION		
1	CATS	contr	w1	▼ Контрагенти		<input type="checkbox"/>
2	CATS1	contract	w2	▼ Договір контрагента		<input type="checkbox"/>
3	number	sumdoc	v1	▼ Сума документа		<input type="checkbox"/>

Scr44

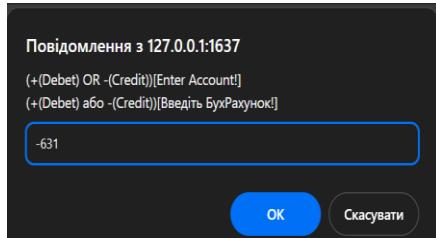
Автоматичне формування рухів

На основі цього рядка програма автоматично будуватиме рухи під час запису документа в Підприємстві.

Скільки таких рядків, розділених символами «;;», стільки й буде сформовано рухів реєстрів для відповідної вкладки (**TITLE** або вибраної **TABLES**).

Рухи по реєстру «account:БухРахунки»

Тут є невелика відмінність.



Scr45

Спочатку також пишемо «+» або «-», але одразу після символу (без пробілу!) вказуємо рахунок, за яким передбачається формування руху.

Із довідника рахунків (у системі Підприємство) будуть братися дані для автоматичного формування рядка, що описує бухРухи.

Приблизно це виглядає так:

```
BALANSE: account:БухРахунки
account/true/w1=accounts#1/w2=nomencl/w3=store/v1=sumcost/v2=kount/vv=act=TMЦ+;/;
account/false/w1=accounts/w2=contr/w3=contract/v1=sumcost/vv=act=КОНТР-/;;
account/true/w1=!6432!/w2=contr/w3=contract/v1=sumvat/vv=act=ПДВ+;/;
account/false/w1=accounts/w2=contr/w3=contract/v1=sumvat/vv=act=КОНТР-/;;
```

Scr46

Вже вище я описував, що означає **accounts** і **accounts#1**, повторюватися не буду. Але що стосується рухів реєстрів, уточню. Коли програма під час запису документа построчно читає рядки в табличних частинах, то одночасно вона бачить усю «шапку» документа, але інші табличні частини (якщо вони є) не бачить. Тому, якщо в «шапці» документа є якийсь ідентифікатор — у даному випадку **accounts**, то **accounts#1** (будь-який **#n**) може бути в кожній табличній частині, і вони жодним чином не конкуруватимуть.

У момент створення документа в Конструкторі програма сама подбає про те, щоб правильно розставити префікси **#n**. А от при описі рухів реєстрів програма може іноді неправильно розставити ідентифікатори з **#n**. Тому довіряй, але перевіряй (штучний інтелект теж помилляється)! Саме для цього у вас є можливість підкоригувати те, що створила програма.

А тепер розберемо, що означає «**!6432!**». Іноді ми можемо напевне знати бухгалтерський рахунок для даного руху. І тоді навіщо забивати документ зайвою колонкою з посиланням на рахунок? Просто ми можемо цей рахунок прописати безпосередньо в описі рухів, обрамлюючи його знаками оклику.

Тут **ТМЦ+ (Прихід)** — це короткий опис операції. Я таке придумав, але ви можете написати власне. У системі **Підприємство** цей рядок буде використано у друкованій формі **BALANSE**, яка є в кожному документі та показує всі рухи реєстрів залишків документа.

Прибуткова накладна #000000000006 > 2025-07-03 20:18:57.000

БухРахунки(account)

w1/w2/w3/w4/w5	v1/v2/v3/v4/v5	ACT
281 Товари на складі / Кофе Totti Supremo / Головний склад	14400/120/0/0/0	ТМЦ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-14400/0/0/0/0	КОНТР-
6432 Податкові зобов'язання непідтвержені / БестТрейд / Договір №258 від 20.10.2019	2880/0/0/0/0	ПДВ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-2880/0/0/0/0	КОНТР-
281 Товари на складі / Замінник молока / Головний склад	40000/500/0/0/0	ТМЦ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-40000/0/0/0/0	КОНТР-
6432 Податкові зобов'язання непідтвержені / БестТрейд / Договір №258 від 20.10.2019	8000/0/0/0/0	ПДВ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-8000/0/0/0/0	КОНТР-
281 Товари на складі / Горячий Поколад / Головний склад	15000/150/0/0/0	ТМЦ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-15000/0/0/0/0	КОНТР-
6432 Податкові зобов'язання непідтвержені / БестТрейд / Договір №258 від 20.10.2019	3000/0/0/0/0	ПДВ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-3000/0/0/0/0	КОНТР-
281 Товари на складі / Кон'як "Наполеон" / Головний склад	400/2/0/0/0	ТМЦ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-400/0/0/0/0	КОНТР-
6432 Податкові зобов'язання непідтвержені / БестТрейд / Договір №258 від 20.10.2019	80/0/0/0/0	ПДВ+
631 Розрахунки з вітчизняними постачальниками / БестТрейд / Договір №258 від 20.10.2019	-80/0/0/0/0	КОНТР-

Scr47

Кореспонденційні рахунки(corr_account)

w1/w2/w3/w4/w5	v1/v2/v3/v4/v5	ACT
281 Товари на складі / 631 Розрахунки з вітчизняними постачальниками	14400/0/0/0/0	
6432 Податкові зобов'язання непідтвержені / 631 Розрахунки з вітчизняними постачальниками	2880/0/0/0/0	
281 Товари на складі / 631 Розрахунки з вітчизняними постачальниками	40000/0/0/0/0	
6432 Податкові зобов'язання непідтвержені / 631 Розрахунки з вітчизняними постачальниками	8000/0/0/0/0	
281 Товари на складі / 631 Розрахунки з вітчизняними постачальниками	15000/0/0/0/0	
6432 Податкові зобов'язання непідтвержені / 631 Розрахунки з вітчизняними постачальниками	3000/0/0/0/0	
281 Товари на складі / 631 Розрахунки з вітчизняними постачальниками	400/0/0/0/0	
6432 Податкові зобов'язання непідтвержені / 631 Розрахунки з вітчизняними постачальниками	80/0/0/0/0	

Рухи грошей по контрагентах(contr_p_m)

w1/w2/w3/w4/w5	v1/v2/v3/v4/v5	ACT
БестТрейд / Договір №258 від 20.10.2019	-83760/0/0/0/0	

Scr48

Поки **ACT** працює лише для бухгалтерії.

Витрату у мене завжди позначається знаком мінус — і не тільки в звіті, але й у таблицях **firebird**!

При згортанні даних не доведеться викручуватися.

❖ Робота механізму створення документа на підставі в системі «Підприємство»

Тепер опишемо, як працює механізм введення **на підставі (IS THE BASIS FOR)**. Я постарався максимально автоматизувати цей процес.

Якщо ми вважаємо, що на підставі редагованого документа потрібно сформувати інший документ, то ставимо галочку після **IS THE BASIS FOR**: — і відкриється вікно з списком усіх документів, які є в базі.

Далі виберіть один або кілька документів (при натиснутій клавіші **Shift**), а потім закрійте галочку — і ваш документ(и) потраплять у текстове поле.



Scr49

Якщо потрібно буде відкоригувати дані — клацніть галочку та виберіть документи знову. На цьому ваша задача в Конструкторі буде завершена.

Реалізація та роль програміста

Коротко поясню, як це реалізовано і яку роль тут відіграє програміст.

Якщо структури вихідного та нового документів схожі (включно зі збіgom ідентифікаторів), система автоматично сформує документ коректно. Можливо, доведеться вручну заповнити один-два елементи.

Однак у випадку, коли структура відрізняється (наприклад, при створенні податкової накладної), необхідне втручання програміста. Для цього передбачені всі умови, щоб ваша робота була ефективною.

■ Перевірка документа при відкритті

При відкритті документа система перевіряє, чи створюється він на підставі іншого.

У **window.onload = function()** використовується такий код:

```
if (myForm.e_title.value === "BASIS") {  
  
    if (typeof MyFunctBASIS == "function") {  
  
        MyFunctBASIS();  
  
    } else doLoadDoc();  
  
}
```

Перевірка функції **MyFunctBASIS**

Програма перевіряє, чи є така функція (**MyFunctBASIS**) у контексті документа.

За замовчуванням такої функції немає, і програма намагатиметься створити документ «на підставі» як вийде.

MyFunctBASIS — це зарезервоване ім'я функції.

Кожна форма об'єкта має можливість підключити один або кілька модулів, які будуть включені в контекст цього об'єкта за допомогою **SRC**.

```
<script src="Myjsjs/DOCS/id_n__/my.js"></script>
```

my.js — це повністю ваш модуль, і ви відповідаєте за його правильну роботу.

Повернемося до документа «на підставі». Якщо у вашому модулі буде функція **MyFunctBASIS**, то вона виконуватиметься, і за її допомогою ви зможете маніпулювати документом у момент його завантаження.

Для правильного розуміння, як з цим впоратися, подивіться, як це реалізовано у функції **doLoadDoc** у модулі **All_db.js**, що стосується «BASIS»!

Таких базових функцій, як **MyFunctBASIS**, у мене буде не одна — це дозволить вам маніпулювати об'єктом як на початку відкриття чи запису, так і наприкінці. Але це я опишу далі.

Якщо ви краще ознайомитеся з **My4s**, то зможете самостійно створювати такі функції та описувати їхню обробку. Це ж не програма, де основна частина записана у закритих файлах (**dll, exe та інших**)!

Про 4SON

Тепер настав час мені зінатися, що спочатку я трохи сказав неправду про **JSON** у полях **blob**.

Якби я взяв за основу базу **PostgreSQL**, то, швидше за все, так би й було. Але я взяв за основу **Firebird**, де вбудована підтримка **JSON** з'явилася лише у версії 5, а я починав з третьої.

Тому я придумав власний метод збереження даних у полях **blob** та їх вилучення звідти. Це — рядок з роздільниками.

Записати **JSON** у поле **blob** не проблема (**JSON.stringify**), а от отримати дані звідти у **Select** виявилося досить складно. Потрібно було написати функцію, аналогічну **JSON.parse**.

Я навіть спробував це зробити вже після того, як у мене все працювало з строкою з роздільниками (де я, а де **JSON**!), і вже функція **Get_Blob** працювала. Але коли я почав писати свою функцію (**Get_JSON**), то зрозумів, що швидкість отримання даних з **JSON** буде у рази більша, ніж з строки з роздільниками. Адже вона використовується у всіх **Select** і неодноразово!

Тому я вирішив залишити все по-старому, хоча й розумів, що **JSON** для тих, хто читає цей текст, був би одразу зрозумілій, і мені не довелося б пояснювати, як я зараз намагаюся це робити.

Далі йде невеликий приклад коду:

```
let strselect_ = "Select GET_BLOB(vv,'t_name') as acc, GET_BLOB(vv,'t_name_') as acc1 From  
MDMD WHERE id = " + select_one;  
  
strselect_ = strselect_.replace("MDMD", t.title.split("|")[0]);  
  
strselect_ = strselect_.replace("t_name", t_name).replace("t_name", t_name);
```

Функція **GET_BLOB** у **Firebird**

Якщо поки вам не все зрозуміло (приклад вирвано з контексту), головне знати, що у **Firebird** є така функція, як **GET_BLOB(vv,'t_name')**.

- **vv** — це поле типу **blob**
- '**t_name'** — ім'я параметра у рядку **vv**, який ми шукаємо

Може бути й третій параметр — **separator** (роздільник). Якщо його немає, то береться роздільник за замовчуванням (у мене роздільник може складатися від одного до трьох будь-яких символів!).

Мені поки вистачило всього трьох рівнів і, відповідно, трьох роздільників.

```
const spr1 = "1~@", spr2 = "2~@", spr3 = "3~@";
```

Структура **4SON**

Перший «**1~@**» завжди йде за замовчуванням!

А тепер перейдемо до опису рядка з роздільниками — назовемо його **4SON**. У рядку може бути багато вкладень з своїми роздільниками.

Ось структура **4SON** з двома рівнями:

```
<name1=param1[=name11=param11spr2name12=param12spr2...]spr1name2=param2[=name21=param21spr2  
name22=param22spr2...]spr1...>
```

Не лякайтесь, усе виявиться набагато простіше!

Головне зрозуміти: **немає жодних пробілів**, а от у параметрах може бути будь-який текст.

У тексті не може бути роздільника, і текст не обрамлюється лапками, якщо цього не вимагає сам текст.

Який тип має рядок (рядок чи число) — при написанні **Select** потрібно вже знати або програмно визначити тип.

Якщо це число, то в **Select** треба буде використати функцію **GET_BLOB4**, яка поверне число.

Я придумав такі роздільники, щоб у реальному тексті була мінімальна ймовірність їх появи. Крім цього, програма в **Підприємстві** жорстко контролює введений операторами текст на наявність цих роздільників!

Простий рядок з одним рівнем

Для початку розглянемо простий рядок з одним рівнем з програми:

«**nomencl=8001~@nomencl_=Замінник молока1~@store#1=7771~@store#1_=Головний склад1~@»**

Трохи відхилюся від теми й поясню, що означають «**nomencl**» та «**nomencl_**».

- Ідентифікатор **без останнього символу нижньої риски** (як назвали у Конструкторі) — це **id** довідника (пам'ятаєте, id унікальний для будь-якого довідника: «**nomencl=800**»).
- А той самий ідентифікатор **з нижньою рискою в кінці** — це вже готове найменування цього елемента довідника.

У деяких звітах це може значно прискорити виконання запиту (**Select...**).

Повертаємося до 4SON

Для прикладу візьмемо рядок з простими односимвольними роздільниками, який знаходиться у полі **blob vv** з трьома рівнями:

a=000*aa=b1=111/b2=222/bb=c1=444&c2=555& /*a1=333*

- Роздільник першого рівня — «*»
- Роздільник другого рівня — «/»
- Роздільник третього рівня — «&»

Отримання рядків першого рівня

Все просто:

GET_BLOB(vv, 'a', '*') = 000

Аналогічно можна отримати рядок:

GET_BLOB(vv, 'a1', '*') = 333

Отримання рядків другого рівня

Спочатку отримуємо рядок, у якому знаходиться другий рівень:

GET_BLOB(vv, 'aa', '*') = b1=111/b2=222/bb=c1=444&c2=555&/

Тут можемо отримати рядок:

GET_BLOB(GET_BLOB(vv, 'aa', '*'), 'b2', '/') = 222

Отримання рядків третього рівня

Щоб отримати **c1** та **c2** (це вже третій рівень з роздільником «&»), спочатку отримуємо рядок «**bb**», у якому знаходиться третій рівень:

GET_BLOB(GET_BLOB(GET_BLOB(vv, 'aa', '*'), 'bb', '/')) = c1=444&c2=555&/

Далі:

GET_BLOB(GET_BLOB(GET_BLOB(GET_BLOB(vv, 'aa', '*'), 'bb', '/'), 'c1', '&')) = 444

GET_BLOB(GET_BLOB(GET_BLOB(vv, 'aa', '*'), 'bb', '/'), 'c2', '&') = 555

Функція GET_BLOB у основній програмі (JS)

Така ж функція **GET_BLOB**, як і у **Firebird**, є й в основній програмі, написаній на **JavaScript**.

Ось приклад, який наочно показує все те, що я намагався пояснити трохи вище:

```
let vv = "a=000*aa=b1=111/b2=222/bb=c1=444&c2=555& /*a1=333*";

let a = GET_BLOB(vv, 'a', '*'); //000

let a1 = GET_BLOB(vv, 'a1', '*'); //333

let aa = GET_BLOB(vv, 'aa', '*'); //b1=111/b2=222/bb=c1=444&c2=555&/

let b1 = GET_BLOB(aa, 'b1', '/'); //111

let b2 = GET_BLOB(aa, 'b2', '/'); //222

let bb = GET_BLOB(aa, 'bb', '/'); //c1=444&c2=555&

let c1 = GET_BLOB(bb, 'c1', '&'); //444
```

```
let c2 = GET_BLOB(bb, 'c2', '&'); //555  
//А можно c2 получить одной строкой:  
  
c2 = GET_BLOB(GET_BLOB(GET_BLOB(vv, 'aa', '*'), 'bb', '/'), 'c2', '&'); //555  
  
//И даже так можно, если вы уверены, что 'bb=' один в строке  
  
c2 = GET_BLOB(vv.split("bb=")[1], 'c2', '&'); //555
```

Про знання 4SON

Взагалі-то вам можна й не знати, як працює **4SON**, і програма сама буде працювати правильно. Але я вважаю, що програмісти, які бажають доопрацювати програму, повинні це знати.

Я чудово розумію, що моя програма ще далека від ідеалу.

Тому я буду радий, якщо мені допоможуть удосконалити програму або хоча б покращати її для себе.

📁 Опис структури файлів проекту на Node.js

Переходимо до опису файлів, на базі яких працює **Node.js**. Усі вони знаходяться на сервері (у демо-версії — швидше за все на вашому комп’ютері) в папці **My4s**. Місце розташування цієї папки ви обираєте самостійно.

Поки що опустимо опис файлів, які знаходяться в корені — частина з них уже згадувалася, інші будуть розглянуті пізніше. Почнемо з папок:

📁 .vscode

Містить файл **launch.json**, який вказує шлях до **node.exe** під час запуску застосунку через **Run** у **VSCode**.

- Якщо **Node.js** уже встановлений на сервері, все працюватиме коректно.
- Якщо ви використовуєте лише файл **node.exe** (наприклад, з кореня **My4s**), необхідно розкоментувати рядок:

```
"runtimeExecutable": "c:/My/My4s/node.exe",
```

Необхідно вказати повний шлях до **node.exe**.

```
{  
  
    // Используйте IntelliSense, чтобы узнать о возможных атрибутах.  
  
    // Наведите указатель мыши, чтобы просмотреть описания существующих атрибутов.  
  
    // Для получения дополнительной информации посетите:  
https://go.microsoft.com/fwlink/?LinkId=830387  
  
    "version": "0.2.0",  
  
    "configurations": [  
  
        {  
  
            "type": "node",  
  
            "request": "launch",  
  
            "name": "Launch App",  
  
            "skipFiles": ["node_modules/**"],  
  
            "preLaunchTask": "build",  
  
            "args": [  
                "app",  
                "arg1",  
                "arg2"  
            ],  
  
            "env": {  
                "NODE_ENV": "development"  
            },  
  
            "localRoot": "app",  
  
            "remoteRoot": null,  
  
            "port": 9229,  
  
            "internalConsoleOptions": "open",  
  
            "breakOnLoad": false,  
  
            "sourceMaps": true,  
  
            "outDir": null,  
  
            "cwd": null  
        }  
    ]  
}
```

```

        "name": "Launch Program",
        // "runtimeExecutable": "c:/My/My4s/node.exe",
        "program": "${workspaceFolder}\\Start4s.js",
    }
]
}

```

Структура папок проєкту

- **MySnipp.code-snippets** — Дуже корисний файл, дозволяє налаштовувати власні випадаючі підказки (**IntelliSense**) у **VSCode**.
- **cf** — Містить шаблони файлів **.htm** для роботи в Конструкторі.
- **db** — Шаблони **.htm** для роботи в Підприємстві.
- **css** — Стилі **.css**. Поки використовується один файл — **TabFix.css**.
- **js** — Скрипти **.js**, що підключаються в HTML-файлах через **src**. Використовуються тільки на клієнті.
- **node_modules** — Містить модулі **.js**, які застосовуються при маршрутизації. Використовуються тільки на сервері.

■ **qq** - Папка порожня, але використовується для обміну інформацією між **Firebird** та **Node.js**.

Статичні файли

Усі перелічені вище папки та файли — **статичні**. Вони працюють з будь-якою базою даних, яких на сервері може бути кілька.

Приклад рядка підключення у браузері

127.0.0.1:1637/mydb/cf

Тут **mydb** — ім'я бази, до якої відбувається підключення. Воно повинно збігатися з назвою папки всередині **My4s**.

У мене це **mydb**, але ви можете задати власне ім'я.

Структура папок бази даних

■ **MyBase.json** – знаходиться в корені папки бази. Містить основні налаштування:

```
{
    "version_cst": "1.0.0", //Версія Конструктора
    "name": "mydb",
    "DSN": "mydb9", //имя DSN підєднання до драйверу ODBC Firebird як налаштуете в windows
    "FDB": "C:/fb/db4s/MYDB9.FDB", //повний шлях до бази на сервері
    "FBK": "C:/fb/db4s/MYDB9.FBK", // повний шлях до файлу весар на сервері
    "Logs": "C:/My/My4S/mydb/logs/", // повний шлях до папки Logs на сервері конкретної бази
    "AllLogs": "C:/My/My/logs_mydb/", // повний шлях до папки AllLogs,де зберігаються старі logs
    "FB": "'c:/Program Files/Firebird/Firebird_3_0/'" // повний шлях до папки Firebird
}
```

На даний момент використовуються лише **"version_cst"** і **"DSN"**. Решта параметрів — на майбутнє.

■ *Автоматично генеровані *.js файли

Під час запису метаданих у Конфігураторі програма автоматично формує або перезаписує *.js файли.

Вони містять готові запити SELECT... (Ajax) та підключаються до HTML-форм за допомогою src:

і файли зберігаються на клієнті в браузері, що значно пришвидшує роботу з базою.

```
<script src="Mydbdb/BalansXX.js"></script>  
<script src="Mydbdb/AccountXX.js"></script>  
<script src="Mydbdb/ID_DBXX.js"></script>
```

📁 Внутрішні папки бази

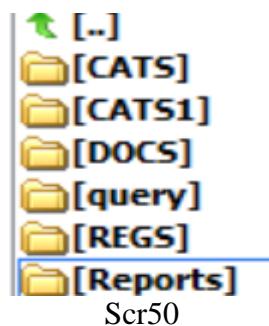
- 📁 Папка **sf** – шаблони .htm та інші файли для роботи в Конструкторі.
- 📁 **Prog.v.us** – JSON-файл з налаштуваннями користувачів для Конструктора. Описувався вище.
- 📁 **Pass.htm** і **Pass2.htm** – шаблони для введення та підтвердження пароля. Універсальні для будь-якої бази, але за бажання можна адаптувати.
- 📁 **index.js** – файл маршрутизації. Підходить для будь-якої бази.
- 📁 **Conf4S.htm** – основна форма, що відкривається у браузері. Універсальна форма для всіх баз.

📁 Папка **db** – шаблони .htm та інші файли для роботи в Підприємстві.

Усі файли в корені цієї папки підходять для роботи з кожною базою.

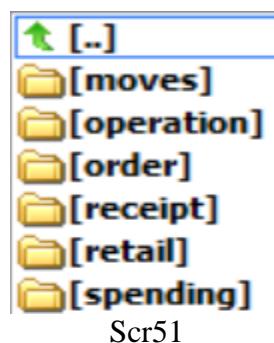
А тепер розберемо папку **Myjsjs**

Усі дані в цій папці формуються автоматично під час роботи з метаданими в Конструкторі.
Папка багаторівнева. Спочатку йдуть усі основні метадані:



І в кожній папці знаходяться папки з назвами всіх ідентифікаторів метаданих, підпорядкованих основній папці.

Розглянемо поки одну **DOCS**:



Імена папок відповідають моїй демо-версії бази (**mydb**). У вашій базі можуть бути інші!

У кожній такій папці обов'язково буде один файл з однаковою назвою **myXX.js**.

У цьому файлі завжди буде один рядок:

`<xml_trXX='[77,68,61,68,79,67,83,47....]'>`

Він формується автоматично, і його не слід коригувати!

Якщо все ж випадково змінили цей файл і метадані перестали правильно завантажуватися, потрібно залити в Конфігуратор у форму, де описується метадані, увімкнути й вимкнути галочку **ACTUAL**, а потім знову записати (**Save**) — і файл буде відновлено.

Змінна **xml_trXX** зберігає готові дані, ніби отримані з бази даних **SELECT... (Ajax)**.

Числа через кому — це коди символів у кодуванні бази **Firebird** (у мене в демо "windows-1251").

Далі є функція перекодування в **UTF-8** — **loadXMLString(xml_trXX)**.

Завдяки цим даним під час відкриття форми «на льоту» буде дописуватися *.htm шаблон, після чого його можна заповнювати даними.

Усе так само, як я описував файли *.js у кореневій папці бази.

Тільки цей файл жорстко прив'язаний до метаданого.

У цю ж папку можна записувати **js-модулі**, у яких ви будете доповнювати потрібну функціональність при роботі з цим метаданим, не ламаючи початкові налаштування.

Цей файл(и) ви будете підключати до шаблону метаданого за допомогою **src**.

Як у мене працює src:

Настав час розповісти, як у мене працює **src**. Покажу на прикладі файлу шаблону **DOCS_db.htm**:

```
<script src="Blob.js"></script>
<script src="All_db.js"></script>
<script src="Find4s.js"></script>
<script src="Formuls.js"></script>
<script src="print.js"></script>
<script src="Myjsjs/DOCS/id_n__/myXX.js"></script>
<script src="Mydbdb/BalansXX.js"></script>
<script src="Mydbdb/AccountXX.js"></script>
<script src="Mydbdb/ID_DBXX.js"></script>
<script src="Myjsjs/DOCS/id_n__/my.js"></script>
```

Програма налаштована так, щоб правильно розпізнавати маршрути до файлів **src**.

Я використовую три види підключення **src**:

1. Пряме зазначення імені файлу .js.

Програма завжди шукає у папці **js**.

2. Пошук файлу в папці конкретного метаданого.

```
src="Myjsjs/DOCS/id_n__/myXX.js"></
```

Оскільки шаблон використовується один для багатьох метаданих, потрібно вказати, де шукати файл *.js.

Для цього використовується мітка **id_n__**, у яку під час підготовки відправки форми з сервера на клієнт буде підставлено ім'я ідентифікатора потрібного метаданого.

Таким чином можна підключити не лише файл **myXX.js**, а й будь-який інший файл, що буде в цій папці (можливо, написаний вами).

3. Підключення файлів, які знаходяться в корені папки з ім'ям бази, з якою ви працюєте.

Mydbdb – це також мітка, на місце якої буде підставлено ім'я бази!

Як ви вже зрозуміли, для кожного підключення можна підставити у ці папки власні файли й потім використовувати їх під час роботи з відповідною формою.

І тепер дуже важлива інформація! При підключенні цих файлів у них можуть повторюватися однакові імена функцій. При виклику такої функції в коді програми виконається та функція, яка знаходиться нижче у підключені **src**!

Досвідчені програмісти вже зрозуміли ідею. Для решти, хто тільки навчається, поясню. Припустимо, у вас уже є написана кимось програма, але вам хочеться щось у ній підкоригувати, при цьому не зламавши ваш код під час оновлення програми. Ви пишете свою функцію в модулі **My.js**, або копіюєте функцію з іншого модуля, яку хочете доопрацювати, і підключаєте ваш модуль у **src** останнім!

Тепер підсумуємо. У папку **js** я б радив вкладати ті власні модулі, які будуть використовуватися в кількох HTML-формах, а прив'язані до одного метаданого — у папку цього метаданого, де вже є файл **«myXX.js»**.

Звіти в My4s Обортно-Сальдова відомість

Обортно-Сальдова відомість — універсальний звіт для всіх регистрів залишків. Він не потребує налаштування в Конструкторі, адже вже вбудований у систему «Підприємство» та автоматично адаптується під параметри кожного регистра.

#	NAME	PRESENTATION	ACTUAL
1	account	БухРахунки	✓
2	contr_p_m	Рухи грошей по контрагентах	✓
3	corr_account	Кореспонденційні рахунки	✓

Scr52

■ Вибір регистра.

У списку відображаються всі регистри залишків, доступні в базі.

Для прикладу відкриємо звіт по регистру **contr_p_m** подвійним клапанням.

Scr53

■ Основні параметри

- **DATE BETWEEN** — задаються дати початку та кінця періоду.
- **Settings** — керування збереженням і завантаженням налаштувань звіту (розглянемо нижче).

■ Верхня таблиця: Виміри

- Відображає всі виміри, налаштовані в Конструкторі.
- Галочка **Choice** — включає вимір у звіт.
- Остання галочка — дозволяє змінювати порядок вимірів, впливаючи на ієархію звіту (верхні — головні, нижні — підпорядковані).

■ Ресурси та форматування

- Нижче — список ресурсів з галочками.
- Можна задати кількість знаків після коми (округлення за потреби).

Scr54

■ Нижня таблиця: Відбори

- Містить рядки умов. У звіт потраплять лише ті записи, які відповідають усім рядкам (логіка AND).
- Галочка **Choice** — включає рядок у план відбору.
- **Wn_Vn** — випадаючий список усіх вимірів і ресурсів.

□ Підпорядковані довідники (наприклад, «Договори») поки що не підтримуються у відборах.



Scr55

?] Приклади роботи з відборами

↗ Відбір за вимірами (наприклад, «Контрагенти»):

- Comparison operators:

- = — один контрагент.
- <> — виключити одного.
- In (— вибрати список контрагентів.
- <>In (— виключити список.
- **VALUE** — не редагується напряму. Подвійне клацання відкриває вікно **FIND** для вибору.

		ADD NEW ROW (Alt+)		DELETE ROW (Alt-)	
Choice	Wn_Vn	Comparison operators	VALUE		
<input checked="" type="checkbox"/>	Контрагенти	In(Бест Трейд, Рабко Сергій Георгієвич, Толокін Микола Михайлович ПП	806,810,808	<input type="checkbox"/>

Scr56

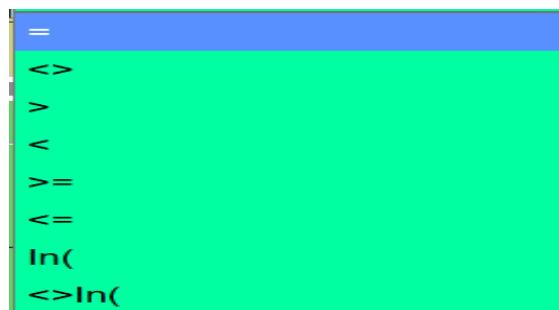
Уточню: у колонці **VALUE** будуть показані найменування контрагентів (через кому), але в запит потраплять лише їхні унікальні **ID**, які зберігаються в **TITLE** даної комірки. Побачити їх можна (хоча нам це й не обов'язково знати), якщо навести курсор на комірку — з'являться: **806, 810, 808**.

Усе так само працюватиме з будь-яким довідником!

У даному прикладі у нас ресурс один, але в інших реєстрах їх може бути більше (до п'яти), і за кожним можна зробити відбір, додавши новий рядок (**ADD NEW ROW (Alt+)**).

Тепер розберемо, як працювати з відбором за ресурсами.

Припустимо, вибрали ресурс «Сума документа», тоді в колонці **Comparison operators** отримаємо інший список.



Scr57

Нагадаю: у **My4s** ресурси — це число.

VALUE — це рядок, який можна редагувати, тобто у простому випадку — це число.

- Якщо захотите вирізати числовий діапазон — введіть в одному рядку «>» число, а потім додайте новий рядок відбору «<».

- Якщо **In** — пишемо кілька чисел через кому (роздільник десяткових — завжди крапка).

Якщо комусь не зрозуміло — ви розберетеся, коли з'явиться хоча б моя демо-версія.

Але це ще все: для відборів з ресурсами є можливість відбирати за оборотом — пишете число, і відбір буде йти лише по колонці оборотів **end-begin**.

- Якщо хочете отримати відбір за початком періоду (**begin**), то перед числом поставте латинську літеру **b** — наприклад: **b250.75**.
- Якщо хочете отримати відбір за кінцем періоду (**end**), то перед числом поставте латинську літеру **e** — наприклад: **e-28550.968** (число менше нуля!).

Знак мінус завжди ставте, якщо число менше нуля.

Ось приклад звіту з відборами:

А ось приклад друкованої форми звіту з відборами:

Рухи грошей по контрагентах			
DATE BETWEEN: 2025-03-01 00:00:00 - 2025-07-31 23:59:59.999 Контрагенти = БестТрейд Сума документа = -83760		Сума документа	
Контрагенти / Договір контрагента		begin	end-begin
BестТрейд		100000.0000	-83760.0000
Договір №258 від 20.10.2019		100000.0000	-83760.0000
TOTAL		100000.0000	-83760.0000
		0 / 83760.0000	16240.0000

Scr58

Зверніть увагу на темний рядок: **0 / 83760.0000** — справа в тому, що в колонці оборотів **end-begin** зберігається не лише оборот, але й приховані при другі, проте доступні для перегляду на комп’ютері дані (знову ж таки використовується **TITLE** при наведенні курсора на потрібну вам цифру). Ви побачите два числа, розділені прямою косою.

- Перше число — прихід (**DEBIT**) за цим рядком за вибраний період.
- Друге число — витрата (**CREDIT**).

Головне, що за цими числами також можна робити відбори!

Просто перед числом поставте латинську літеру:

- d** — якщо потрібен прихід (**DEBIT**, наприклад **d0**).
- c** — витрата (**CREDIT**, наприклад **c83760**).

Усі звіти в **My4s** підтримують розшифровку за оборотами. Якщо зробити подвійне клацання на обороті (**end-begin**), відкриється список усіх документів, які рухали цю цифру за вибраний період.

METADATA: DOCS		USER:demo	DOCUMENT: AllDOCS	
#	DOCUMENT	ACT.	INFORMATION	
1	Прибуткова накладна #000000000006 > 2025-07-03 20:18:57.000	✓	БестТрейд Головний склад 83760.0000	
2	Прибуткова накладна #000000000004 > 2025-03-02 15:03:04.241	✓	БестТрейд Головний склад 60225.6000	

Scr59

Далі є можливість — подвійним клацанням відкрити документ, навіть віредагувати його, зберегти та заново запустити звіт.

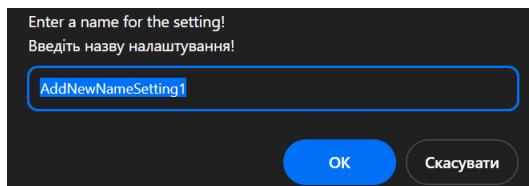
А тепер опишемо, як працює **Settings**.

Settings:	NONE ▾	AddNew	Save	Delete	OpenSetting
-----------	--------	--------	------	--------	-------------

Scr60

Коли зробите якийсь відбір і захочете зберегти його для майбутнього використання — тут вам і стане у пригоді **Settings**.

Натисніть **AddNew** — відкриється вікно.



Scr61

Введіть ім'я вашого налаштування. Воно з'явиться у випадаючому списку.

Наступним кроком натисніть **Save**.

Дані записуться в таблицю Firebird «**SETTING_ALL**».

MD	ID_N	USERS	NAME	ACTUAL	VV
► QUERY	account	demo	281	✓	(BLOB)
QUERY	account	demo	631	✓	(BLOB)
QUERY	account	demo	2821	✓	(BLOB)
Reports	tovar	demo	rr	✓	(BLOB)
Reports	tovar	demo	ww	✓	(BLOB)
Reports	cats_nomencl	demo	kount	✓	(BLOB)
Reports	cats_nomencl	demo	count	✓	(BLOB)
Reports	cats_nomencl	demo	Контрагент	✓	(BLOB)
query	prov1	demo	contr	✓	(BLOB)
QUERY	contr_p_m	demo	contr	✓	(BLOB)
query_fast	prov3	demo	contr	✓	(BLOB)
query_fast	prov5	demo	formul-	✓	(BLOB)
QUERY	contr_p_m	demo	ПримерКонтр	✓	(BLOB)

Scr62

При новому відкритті звіту «**Оборотно-сальдова відомість**» ви можете вибрати своє налаштування з випадаючого списку за ім'ям.

Відобразяться всі збережені вами конфігурації.

Якщо потрібно внести зміни — просто знову натисніть **Save**, і оновлене налаштування буде збережено.

■ Як зберігаються налаштування

Важливо розуміти: при натисканні **Save** налаштування зберігаються не лише в базі даних **Firebird**, але й локально в браузері (через **localStorage**). Звіт у першу чергу працює з даними з браузера. Можливо, ви вже помітили, що я намагаюся мінімізувати звернення до бази даних — усе заради того, щоб система могла комфортно обслуговувати не десятки, а тисячі клієнтів одночасно (а в ідеалі — ще більше!).

□ Очищення браузера та відновлення налаштувань

Іноді браузер доводиться чистити, і тоді ваші локальні налаштування в **Settings** зникають. Але ми ж не дарма зберігаємо їх у базі! Для відновлення є кнопка **OpenSetting** — після її натискання всі ваші налаштування будуть завантажені з бази й знову записані в кеш браузера. Кнопка **OpenSetting** буде прихована, якщо налаштування вже присутні у браузері.

■ Універсальність **Settings**

Налаштування **Settings** доступні у всіх звітах, створених у **My4s**, і працюють практично однаково. Тому далі я їх описувати не буду.

Звіти, створювані в Конструкторі У Конструкторі відкриємо в меню — Звіти (Report):

METADATA:		USER:		
© ADD NEW > >		© Update		
#	NAME	PRESENTATION		ACTUAL
1	cats_nomencl	Номенклатура//Довідники		✓
2	tovar	Товари//Gr1		✓

Scr63

Тут відображаються звіти, створені через меню «**Звіти**».

Щоб створити новий — натисніть **ADDNEW**.

Усі звіти формуються на основі об'єднання кількох таблиць за допомогою **UNION**. Використання **JOIN** також можливе, але в цьому випадку потрібно вручну доопрацювати SQL-запит (**SELECT...**).

Створення нового звіту

При створенні нового звіту необхідно задати:

- **NAME** — унікальне ім'я звіту (зверніть увагу: після збереження змінити його буде складно);
- **PRESENTATION** — відображувана назва.

Далі у полі **TITLE** виберіть з списку **TABLES** потрібні таблиці **Firebird** — стільки разів, скільки необхідно для формування запиту **UNION**.

Після кожного вибору в полі **textarea** автоматично додається рядок такого вигляду:

```
SELECT * FROM DOCS2 WHERE ID_N='idd_` AND DAT BETWEEN dat8_ AND dat9_ AND 1=1;;
```

Після вибору таблиці з'явиться додатковий випадаючий список з можливими значеннями **ID_N** — це потрібно для уточнення, з яким документом працювати (актуально для таблиць **DOCS** та **DOCS2**).

На словах це може звучати складно, але при запуску демо-версії все стає наочним і зрозумілим.

Система працює автоматично — головне, правильно вибрати параметри.

Розглянемо створення звіту на прикладі вже готового звіту **«tovar»**.

Якщо ви не знайомі з принципом роботи **UNION** у SQL, рекомендую спершу ознайомитися з документацією — найкраще по **Firebird**, хоча в інших СУБД механізм працює схожим чином.

Звіт формується лише за документами:

- **Прибуткова накладна**
- **Переміщення**
- **Роздрібна торгівля**

Особливість: документ **Переміщення** використовується двічі, оскільки містить два склади — джерело та отримувач.

При кожному виборі документа в полі **textarea** додаються початкові **SELECT**-запити, розділені символами «**;;**».

Одночасно автоматично створюються шаблони таблиць — поки що порожні, але готові до заповнення.

Edit Metadata:> USER: Программист1 ROLES: zakupki,buh

METADATA: Reports NAME: tovar PRESENTATION: Товари ACTUAL:

Select Form:

TITLE TABLES MaketPrint DESCRIPTION

TABLES: NONE

TITLE TABLES MaketPrint DESCRIPTION

TABLES: NONE

Я вже вище розповідав, що **DOCS2** — це представлення (**VIEW**), яке з'єднує дві таблиці **Firebird** в одну (**DOCS** і **ROWS_DOCS**)!

Тепер переходимо на вкладку **TABLES** і бачимо вже заповнену таблицю №1 з чотирьох.

Але це вже готовий запит — повністю готовий до збереження, тоді як насправді таблиця спочатку буде порожньою.

The screenshot shows the 'Edit Metadata' interface for a table named 'DOCS2_receipt'. At the top, there are fields for 'USER' (Programmist1), 'ROLES' (zakupki.buh), and buttons for 'SAVE METADATA >>' and 'COPY NEW >>'. Below this, there are sections for 'METADATA' (Reports), 'NAME' (товар), 'PRESENTATION' (Товари), and 'ACTUAL: checked'. A 'Select Form:' dropdown is set to 'TITLE'. Below these are buttons for 'TABLES' (selected), 'MarketPrint', 'DESCRIPTION', and page numbers (1, 2, 3, 4). An 'ADD TABLE >>' button is also present. The main area shows the table structure with 13 rows. The columns are: ID, Numb., Name, VV_TAB, Width %, Title, Same, Hidden, ADD Attributes, AS, and a checkmark column. The table contains various formulas like 'store', 'nomencl', 'price', 'kount', 'formul1', etc., and calculated columns like 'store', 'nomencl', 'price', 'kount', 'kount1', '(kount-kount1)', 'sumrow', 'sumrow1', '(sumrow-sumrow1)', 'sumvat', 'sumvat1', '(sumvat-sumvat1)', and 'id_docs'. Buttons for 'ADD NEW ROW (+)' and 'DELETE ROW (-)' are located at the top of the table area.

Scr65

Загальні принципи налаштування звіту

DOCS2_receipt — це ім'я таблиці, автоматично сформоване програмою. Ви можете змінити його, але це жодним чином не вплине на роботу звіту.

Перед початком заповнення таблиць важливо заздалегідь продумати структуру запиту:

- **Таблиця №1** вважається головною — усі інші підлаштовуються під неї.
- Кількість рядків у всіх таблицях завжди однакова.
- Виміри йдуть першими, і параметр **Width** береться з первого рядка. Інші виміри (не ресурси) на **Width** не впливають.
- У колонці **Numb** необхідно поставити галочку для всіх ресурсів.

Робота з колонкою AS

Колонка **AS** використовується аналогічно до SQL-запиту `SELECT ... AS ...`.

Вона заповнюється автоматично, але за потреби ви можете її змінити — у деяких випадках це навіть обов'язково.

Додавання нового рядка

При додаванні нового рядка завжди починайте з поля **Name**. У випадаючому списку перші два параметри — стандартні:

- **Empty** — означає, що в поточній таблиці параметр відсутній, але він є хоча б в одній з інших. Усі інші поля (крім **Name** та **VV_TAB**) все одно потрібно заповнити.
- **Formul** — вказує, що в колонці буде використовуватися формула. У вікні, що з'явиться, потрібно вказати ім'я формулі. Усі формули задаються в **Таблиці №1**.

Для формул колонка **AS** має особливe значення — це сама формула.

Усередині круглих дужок використовуйте ідентифікатори з інших рядків **AS**.

Параметри Name, VV i TAB

Усі інші параметри в **Name** беруться з структури відповідної таблиці **Firebird**.

Особливу увагу слід звернути на параметри **VV** і **TAB** (завжди присутній хоча б один). Це **BLOB**- поля з інформацією у форматі **4SON** (описано вище).

При виборі такого параметра активується колонка **VV_TAB**, у якій можна вибрати конкретні параметри з **BLOB**- поля.

Мова та форматування

- Усі поля заповнюються латиницею, крім колонки **Title** — тут можна використовувати довільний текст.
- Колонка **ADD Attributes** не використовується, як описано вище. Це просто рядок, призначений для додаткових налаштувань при виконанні звіту.

Приклад використання (поки що маю один випадок, але в майбутньому може ще знадобитися):
 Якщо у виміри потрапляє не посилання на довідник, а звичайне число (наприклад, **price**), потрібно вказати програмі, що це число. Для цього в колонці **ADD Attributes** пишемо: «**!number!**».
 Без цього рядка звіт працюватиме, але відбір за ціною виконати не вдасться.

Галочка в колонці **Same** означає, що параметр буде розміщений у тій самій комірці, що й попередній (вище) — тобто, у два рівні (поверхи).

Для інших таблиць головне — налаштувати їй пов’язати рядки з **Таблицею №1** у колонках **Name** та **VV_TAB**.

Якщо зв’язку немає — вибираєте **Empty**.

Інші колонки ролі не грають, незалежно від того, що в них написано!

Select Form:

• TITLE • TABLES • MaketPrint • DESCRIPTION 1 2 3 4 ADD TABLE >>

Name TABLE: DOCS2_moves													COPY	PASTE	DELETE TABLE
													ADD NEW ROW (+)	DELETE ROW (-)	
13	Numb.	Name	VV_TAB	Width %	Title	Same	Hidden	ADD Attributes					AS	✓	
1		TAB	store#1	50		<input type="checkbox"/>	<input type="checkbox"/>						store#1	<input checked="" type="checkbox"/>	
2		TAB	nomenc1	50		<input type="checkbox"/>	<input type="checkbox"/>						nomenc1	<input checked="" type="checkbox"/>	
3				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
4		TAB	kount	10		<input type="checkbox"/>	<input type="checkbox"/>						kount	<input checked="" type="checkbox"/>	
5				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
6				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
7				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
8				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
9				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
10				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
11				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
12				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
13		ID				<input type="checkbox"/>	<input type="checkbox"/>						id_docs	<input checked="" type="checkbox"/>	

Scr66

Select Form:

• TITLE • TABLES • MaketPrint • DESCRIPTION 1 2 3 4 ADD TABLE >>

Name TABLE: DOCS2_moves1													COPY	PASTE	DELETE TABLE
													ADD NEW ROW (+)	DELETE ROW (-)	
13	Numb.	Name	VV_TAB	Width %	Title	Same	Hidden	ADD Attributes					AS	✓	
1		VV	store			<input type="checkbox"/>	<input type="checkbox"/>						store	<input checked="" type="checkbox"/>	
2		TAB	nomenc1			<input type="checkbox"/>	<input type="checkbox"/>						nomenc1	<input checked="" type="checkbox"/>	
3						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
4		TAB	kount			<input type="checkbox"/>	<input type="checkbox"/>						kount	<input checked="" type="checkbox"/>	
5						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
6						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
7						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
8						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
9						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
10						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
11						<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
12				10	ПДВ(+)	<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
13		ID				<input type="checkbox"/>	<input type="checkbox"/>						id_docs	<input checked="" type="checkbox"/>	

Scr67

Select Form:

• TITLE • TABLES • MaketPrint • DESCRIPTION 1 2 3 4 ADD TABLE >>

Name TABLE: DOCS2_retail													COPY	PASTE	DELETE TABLE
													ADD NEW ROW (+)	DELETE ROW (-)	
13	Numb.	Name	VV_TAB	Width %	Title	Same	Hidden	ADD Attributes					AS	✓	
1		VV	store	50		<input type="checkbox"/>	<input type="checkbox"/>						store	<input checked="" type="checkbox"/>	
2		TAB	nomenc1	50		<input type="checkbox"/>	<input type="checkbox"/>						nomenc1	<input checked="" type="checkbox"/>	
3		TAB	price	10		<input type="checkbox"/>	<input type="checkbox"/>						price	<input checked="" type="checkbox"/>	
4				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
5		TAB	kount	10		<input type="checkbox"/>	<input type="checkbox"/>						kount	<input checked="" type="checkbox"/>	
6				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
7				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
8		TAB	sumrow	10		<input type="checkbox"/>	<input type="checkbox"/>						sumrow	<input checked="" type="checkbox"/>	
9				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
10		TAB	sumvat	10		<input type="checkbox"/>	<input type="checkbox"/>						sumvat	<input checked="" type="checkbox"/>	
11				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
12				10		<input type="checkbox"/>	<input type="checkbox"/>							<input checked="" type="checkbox"/>	
13		ID				<input type="checkbox"/>	<input type="checkbox"/>						id_docs	<input checked="" type="checkbox"/>	

Scr68

Так буде виглядати звіт у Підприємстві:

Товари						
Склад / Номенклатура / Ціна	Кільк(+) Кільк(-)	Кільк(+)	Сум(+) Сум(-)	Сум(+)	ПДВ(+) ПДВ(-)	ПДВ(+)
Головний склад	772.00 0.00	772.00	83760.00 0.00	83760.00	13960.00 0.00	13960.00
Горячий Шоколад	150.00 0.00	150.00	18000.00 0.00	18000.00	3000.00 0.00	3000.00
100	150.00 0.00	150.00	18000.00 0.00	18000.00	3000.00 0.00	3000.00
Замінник молока	500.00 0.00	500.00	48000.00 0.00	48000.00	8000.00 0.00	8000.00
80	500.00 0.00	500.00	48000.00 0.00	48000.00	8000.00 0.00	8000.00
Коп'як "Наполеон"	2.00 0.00	2.00	480.00 0.00	480.00	80.00 0.00	80.00
200	2.00 0.00	2.00	480.00 0.00	480.00	80.00 0.00	80.00
КофеTotti Supremo	120.00 0.00	120.00	17280.00 0.00	17280.00	2880.00 0.00	2880.00
120	120.00 0.00	120.00	17280.00 0.00	17280.00	2880.00 0.00	2880.00
TOTAL:	772.00 0.00	772.00	83760.00 0.00	83760.00	13960.00 0.00	13960.00

Scr69

Як ви вже помітили, у всіх моїх друкованих формах виміри завжди виводяться в одній колонці — першій.

Річ у тім, що запит **SELECT...** повертає дані у вигляді одного рядка, згорнутого за структурою «Склад / Номенклатура / Ціна».

У друкованій формі вони відображаються як окремі рядки з візуальним зміщенням (довгий пробіл).

Спочатку я реалізував запит так, щоб він одразу повертає дані у готовому вигляді — як ви бачите на екрані. Сам запит при цьому ставав значно об'ємнішим, але це не було критично.

Проблема виникла в іншому: сервер **Firebird** починає витрачати суттєво більше часу на виконання такого запиту.

Тому я відмовився від цієї ідеї й переніс логіку розбиття на підпорядковані виміри та обчислення сум за кожним виміром на клієнтську сторону. Так, довелося чимало попрацювати, але код я написав один раз і тепер застосовую його у всіх подібних запитах.

У звіті також реалізовано розшифровку за документами (якщо це звіт по документах). Для цього додається прихована колонка **id_docs**.

Однак розшифровка працює лише за останнім виміром — у даному прикладі це «**Ціна**» (по подвійній дії миші).

Закладка MaketPrint:

Select Form:

TITLE TABLES MaketPrint DESCRIPTION

TABLES: NONE

```
WITH MyW AS (
  SELECT Get_blob_A(vv,'store') AS store,Get_blob_A(tab,'nomencl') AS nomencl,Get_blob_A(tab,'price') AS price,SUM(Get_blob_4(tab,'kount')) AS kount,0 AS kount1,SUM(Get_blob_4(tab,'sumrow')) AS sumrow,0 AS sumrow1,SUM(Get_blob_4(tab,'sumvat')) AS sumvat,0 AS sumvat1
  FROM DOCS2
  WHERE ID_N='receipt' and DAT BETWEEN dat8_ and dat9_ and 1=1 GROUP BY 1,2,3
  UNION ALL
  SELECT Get_blob_A(tab,'store#1'),Get_blob_A(tab,'nomencl'),0,SUM(Get_blob_4(tab,'kount')),0,0,0,0,0,LIST(ID)
  FROM DOCS2
  WHERE ID_N='moves' and DAT BETWEEN dat8_ and dat9_ and 1=1 GROUP BY 1,2,3
  UNION ALL
  SELECT Get_blob_A(vv,'store'),Get_blob_A(tab,'nomencl'),0,SUM(Get_blob_4(tab,'kount')),0,0,0,0,0,LIST(ID)
  FROM DOCS2
  WHERE ID_N='moves' and DAT BETWEEN dat8_ and dat9_ and 1=1 GROUP BY 1,2,3
  UNION ALL
  SELECT Get_blob_A(vv,'store'),Get_blob_A(tab,'nomencl'),Get_blob_A(tab,'price'),0,SUM(Get_blob_4(tab,'kount')),0,SUM(Get_blob_4(tab,'sumrow')),0,SUM(Get_blob_4(tab,'sumvat')),LIST(ID)
  FROM DOCS2
  WHERE ID_N='retail' and DAT BETWEEN dat8_ and dat9_ and 1=1 GROUP BY 1,2,3
)
SELECT store,nomencl,price,SUM(kount) AS kount,SUM(kount) AS kount1,SUM(kount-kount1) AS formul1,SUM(sumrow) AS sumrow,SUM(sumrow1) AS sumrow1,SUM(sumrow-sumrow1) AS formul2,SUM(sumvat) AS sumvat,SUM(sumvat1) AS sumvat1
FROM MyW
WHERE 2=2
GROUP BY 1,2,3
//...
<h3 style="text-align: center;" onclick="mydocdiv.hidden=false;prn.hidden=true">..txtt..</h3><aside style="text-align: center; background-color: yellow">..titl1..</aside>
<tr>
<th style="width: 40%;">Склад / Номенклатура / Ціна</th>
<th style="width: 10%;">Кільк(+)</th>
<th style="width: 10%;">Кільк(-)</th>
<th style="width: 10%;">Сум(+)</th>
<th style="width: 10%;">Сум(-)</th>
<th style="width: 10%;">ПДВ(+)</th>
<th style="width: 10%;">ПДВ(-)</th>
<th style="width: 30%;>id_docs</th>
```

Scr70

Спочатку поле **textarea** залишається порожнім.

Тільки після повного завершення формування всіх таблиць можна натиснути кнопку «**Generate Select and Maket**» — програма автоматично сформує **SQL-запит** та **HTML-код**.

Якщо таблиці заповнені коректно, звіт має успішно запрацювати в системі «**Підприємство**».

Якщо звіт працює, текст можна редагувати — за умови, що ви точно розумієте, що робите.

Втім, ніхто не забороняє поекспериментувати: доопрацювати текст, зберегти та запустити його у «**Підприємство**».

Якщо результат виявиться невдалим — не страшно. Просто видаліть уесь текст і знову натисніть кнопку «Generate Select and Maket».

Зверніть увагу: якщо в **textarea** залишиться хоча б один символ, кнопка працювати не буде.

Коли текст готовий, можна переглянути структуру звіту, натиснувши кнопку «View print».

..txt..						
..fill..						
Склад / Номенклатура / Ціна	Кільк(+) Кільк(-)	Кільк(+-)	Сум(+) Сум(-)	Сум(+-)	ПДВ(+) ПДВ(-)	ПДВ(+-)
..store.. / ..nomencl.. / ..price..	..kount.. ..kount1..	..formul1..	..sumrow.. ..sumrow1..	..formul2..	..sumvat.. ..sumvat1..	..formul3..

Scr71

Як я вже писав — у **Підприємстві** при кліку по шапці друкованої форми (у нашому випадку «Товари») можна повернутися до налаштувань звіту.

Там можна змінити дати чи відбори й знову сформувати звіт.

Також можна повернутися до налаштувань звіту, натиснувши **Esc**, але є невелика відмінність — крім налаштувань звіту залишиться видимою друкована форма трохи нижче налаштувань!

Це працює майже у всіх звітах.

Наступна вкладка **DESCRIPTION**. Це просто поле **textarea**.

На роботу звіту воно не впливає. Туди можна написати щось на кшталт **Help**, або будь-який текст, який вважаєте потрібним.

Наступний звіт — Помічник написання додаткових звітів

У Конструкторі відкриваємо в меню — **Помічник написання додаткових звітів**.

Цей конструктор дозволяє створювати майже такі самі звіти, як і попередній, але час, витрачений на створення, я сподіваюся, буде у кілька разів менший.

Річ у тім, що попередній звіт вимагає доброго розуміння того, як працює **UNION** у запитах (**SELECT**). У цьому ж звіті не потрібно чітко знати й дотримуватися правил побудови звіту з використанням **UNION**.

Достатньо зробити вибірки даних з різних таблиць (можна й з однієї!), а програма вже сама об'єднає дані таблиць за законами **UNION**.

Приступимо до опису створення нового звіту.

Edit Metadata:> USER: demo ROLES:

METADATA: query NAME: PRESENTATION: ACTUAL:

Query (Вільний звіт) Help
ADD_NEW FROM SampleTable Same DATE

 [Online htm editor](#)

Scr72

Початок роботи з новим звітом

- **Шапка документа:** її заповнення описано вище, тому тут не розглядається.
- **Help:** якщо поставити галочку, відкриється просте поле **textarea**, куди можна внести коментар або опис звіту на ваш розсуд.
- Стрілки «вліво/вправо»: використовуються переважно для перевірки роботи запиту наприкінці створення звіту, за допомогою кнопки «RUN>>».

Важливі правила при створенні звіту

1. На початку роботи поле **textarea** завжди має бути порожнім.
Редагувати його не можна доти, доки не будуть заповнені всі необхідні дані.

2. Робота починається з натискання кнопки «ADD_NEW».

Після цього в полі з'явиться службовий текст:

Firebird: Construct a request for such parameters! Побудуємо запрос по таким параметрам://MySelect

- Далі необхідно приступити до вибору таблиць і даних, які будуть використовуватися у звіті.



Scr73

Вибір таблиці для звіту

У випадаючому списку **FROM** необхідно вибрати потрібну таблицю з **Firebird**.

Усі таблиці були описані раніше, але окремо варто пояснити таблицю-представлення **BALANS_DOC**.

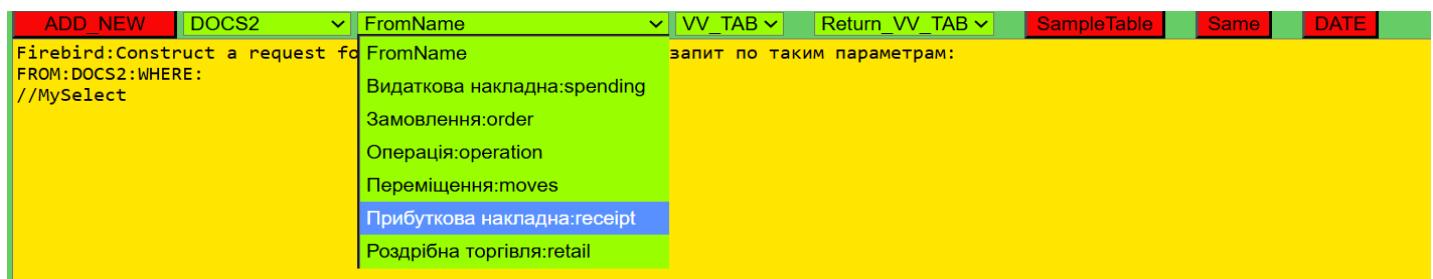
BALANS_DOC — це представлення (**VIEW**), яке надає доступ до всіх рухів документів.

- Докладніше ознайомитися з його структурою можна в редакторі **Firebird**.

Для прикладу виберемо представлення **DOCS2**.

Після цього з'являться додаткові випадаючі списки:

- **FromName**
- **VV_TAB**



Scr74

Робота з випадаючими списками

- У списку **FromName** відображаються всі документи, доступні в базі.

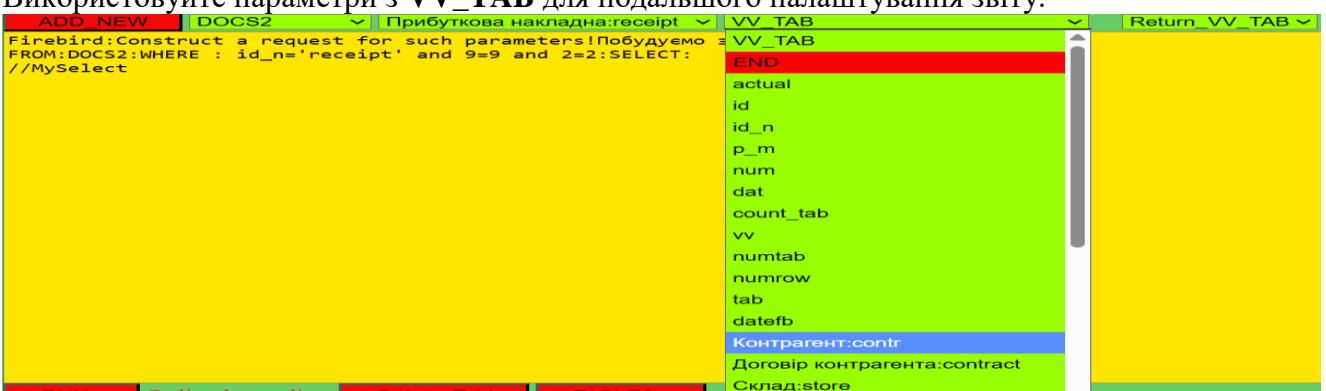
- Для прикладу виберемо документ **receipt** (Прихідна накладна).

Після вибору документа автоматично заповнюється випадаючий список **VV_TAB**.

У ньому будуть представлені всі параметри вибраної таблиці, включно з параметрами, що знаходяться у **Blob- полях** документа.

Таким чином, послідовність дій виглядає так:

1. Виберіть документ у списку **FromName**.
2. Дочекайтесь автоматичного заповнення списку **VV_TAB**.
3. Використовуйте параметри з **VV_TAB** для подальшого налаштування звіту.



4.

Scr75

Ось тут і почнеться справжня робота програміста.

Ви вже повинні бути готовими до того, які параметри вам знадобляться з цього документа.

Тоді послідовно вибирайте ці параметри з випадаючого списку.

Після кожного вибору обрані параметри будуть видалятися з випадаючого списку та з'являтися у спеціальному полі для контролю.



Scr76

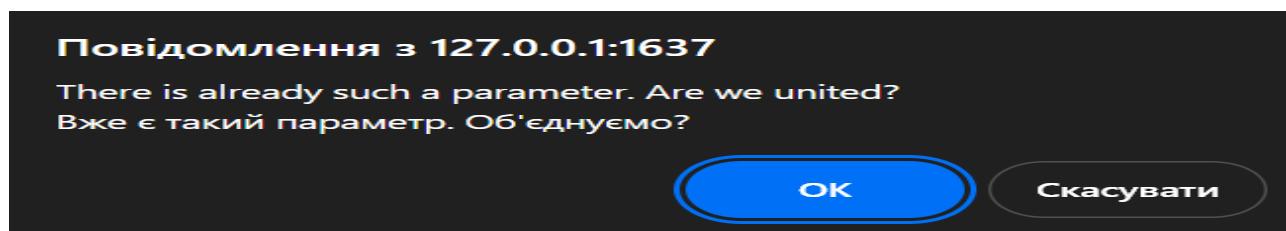
Завершення вибору параметрів.

На цьому етапі важливо бути уважним:

- Якщо допущена помилка, доведеться починати все спочатку.
- Для цього можна обнулити поле **textarea** (Ctrl+A → Delete) або просто перезавантажити сторінку.

Якщо все зроблено правильно:

- Наприкінці всіх виборів необхідно вибрати з списку пункт **END** (виділений червоним кольором). Це сигнал програмі, що етап вибору завершено.
- Після цього в полі **textarea** над рядком //MySelect з'явиться новий рядок з попереднім набором даних, вибраних вами раніше:
- FROM:DOCS2:WHERE : id_n='receipt' and dat BETWEEN dat8_ and dat9_ and 2=2:SELECT :Get_blob_A(vv,'contr') AS contr,Get_blob_A(vv,'store') AS store,SUM(Get_blob_4(tab,'kount')) AS kount,
- □□
- Не лякайтесь — це рядок, який у кінцевому підсумку дозволить програмі правильно сформувати фінальний **SELECT**.
- Якщо запит у вас буде використовувати кілька таблиць, тоді знову йдете у список **FROM**, вибираєте нову таблицю й повторюєте все так само, як з першою таблицею.
- Але тепер є важливий нюанс: ми знову готуємо запит з використанням **UNION**. При кожному виборі параметра програма аналізує ім'я параметра на збіг з іменем з попередніх таблиць, і якщо ім'я збігається — з'являється вікно.



Scr77

Якщо натиснути **OK**, то програма запам'ятає, що ці параметри потрібно об'єднати під час створення вже підсумкового **SELECT** з використанням **UNION**.

Якщо натиснути «**Відмінити (Cancel)**», тоді буде створено новий параметр і з'явиться вікно.



Scr78

Новий параметр отримує ім'я, яке створюється з нової назви та імені таблиці, з'єднаних через символ підкреслення.

Це потрібно для того, щоб програма правильно формувала рядки **SELECT** з використанням **UNION**.

Далі запит буде автоматично за правилами **UNION**.

Є важливий момент: якщо у звіті беруть участь документи й потрібна розшифровка у друкованій формі, то при вибірці параметрів документів останнім обов'язково вказуйте **id документа**.

Після вибірки з таблиць переходимо до останнього кроку — відкриваємо список **Return_VV_TAB**. У ньому, крім стандартних пунктів (**END**, **formul_**, **ALL**), будуть показані всі вибрані параметри в тому порядку, у якому вони були додані.

- Якщо параметри спочатку розташовані правильно (спочатку виміри, потім ресурси) і формули не потрібні, можна вибрати **ALL** — тоді всі параметри автоматично потраплять у рядок **Return**.
- Якщо порядок інший або потрібні формули, параметри доведеться вибирати вручну, по одному, у тому порядку, як вони мають виводитися у друкованій формі (спочатку виміри, потім ресурси).

Щоб вставити формулу в колонку, виберіть параметр **formul_**.

Врахуйте: якщо параметр використовується лише всередині формул і не потрібен у друкованій формі, його вказувати при виборі не потрібно.

Після завершення вибору параметрів виберіть **END**.

У полі **textarea** над рядком `//MySelect` з'явиться новий рядок з попереднім набором даних, вибраних раніше.

Приклад:

```
«Return: Select : contr,store,nomencl,price,SUM(kount) AS kount,SUM(sumvat) AS sumvat,SUM(sumrow) AS sumrow,SUM(kount_spending) AS kount_spending,SUM(sumvat_spending) AS sumvat_spending,SUM(sumrow_spending) AS sumrow_spending, SUM(formul_1) AS formul_1»
```

А після `«//MySelect»` з'явиться вже згенерований програмою **SELECT**, приблизно такого вигляду:

```
«WITH q_all AS (
    SELECT Get_blob_A(vv,'contr') AS contr,Get_blob_A(vv,'store') AS store,Get_blob_A(tab,'nomencl') AS nomencl,Get_blob_T(tab,'price') AS price,SUM(Get_blob_4(tab,'kount')) AS kount,SUM(Get_blob_4(tab,'sumvat')) AS sumvat,SUM(Get_blob_4(tab,'sumrow')) AS sumrow,NULL AS price_spending,NULL AS kount_spending,NULL AS sumvat_spending,NULL AS sumrow_spending
    FROM DOCS2 WHERE id_n='receipt' and dat BETWEEN dat8_ and dat9_ and 2=2
    GROUP BY contr,store,nomencl,price,price_spending
    UNION ALL
    SELECT Get_blob_A(vv,'contr') AS contr,Get_blob_A(vv,'store') AS store,Get_blob_A(tab,'nomencl') AS nomencl,NULL AS price,NULL AS kount,NULL AS sumvat,NULL AS sumrow,Get_blob_T(tab,'price') AS price_spending,SUM(Get_blob_4(tab,'kount')) AS kount_spending,SUM(Get_blob_4(tab,'sumvat')) AS sumvat_spending,SUM(Get_blob_4(tab,'sumrow')) AS sumrow_spending
    FROM DOCS2 WHERE id_n='spending' and dat BETWEEN dat8_ and dat9_ and 2=2
    GROUP BY contr,store,nomencl,price,price_spending
)
SELECT contr,store,nomencl,price,price_spending,SUM(kount) AS kount,SUM(sumvat) AS sumvat,SUM(sumrow) AS sumrow,SUM(kount_spending) AS kount_spending,SUM(sumvat_spending) AS sumvat_spending,SUM(sumrow_spending) AS sumrow_spending SUM(formul_1) AS formul_1
FROM q_all
GROUP BY contr,store,nomencl,price,price_spending;»
```

На етапі доопрацювання формул необхідно відредактувати рядок виду:

SUM(formul_1) AS formul_1

замінивши його на рядок, що відображає потрібний розрахунок.

Наприклад:

SUM(kount - kount_spending) AS formul_1

Кількість формул може бути будь-якою.

Програма автоматично створює їх, використовуючи лічильник: **formul_1**, **formul_2**, **formul_3** і так далі.

Після коректного заповнення всіх параметрів у запиті можна завершити формування блоку **SELECT**.

У цьому випадку запит буде виконано правильно й він поверне очікуваний результат.

Далі сформуємо **HTML-код** для виведення друкованої форми.

Натисніть кнопку “**SampleTable**” — програма сама сформує HTML-код.

Щоб переглянути структуру друкованої форми внизу, натисніть кнопку «**TABLES**» — у нижній частині екрана з’явиться представлення структури друкованої форми.

The screenshot shows the software's interface for creating reports. At the top, there is a code editor window containing the following HTML code:

```
//TAB
<h3 style="text-align: center;" onclick="mydocdiv.hidden=false;prn.hidden=true">..txtt..</h3><aside style="text-align: center; background-color: yellow">..titl1..</aside>
<table id="tab_prn" border="1">
<tr>
  <th width=60%>Контрагент / Склад / Номенклатура / Ціна / Ціна</th>
  <th width=10%>Кількість</th>
  <th width=10%>Сума ПДВ</th>
  <th width=10%>Вся сума</th>
  <th width=10%>Кількість1</th>
  <th width=10%>Сума ПДВ1</th>
  <th width=10%>Вся сума1</th>
</tr>
<tr>
  <td ondblclick="Dblclick_DOC(this)" style="text-align:left;">..contr.. / ..store.. / ..nomencl.. / ..price.. / ..price_spending..</td>
  <td style="text-align:right;">..kount..</td>
  <td style="text-align:right;">..sumvat..</td>
  <td style="text-align:right;">..sumrow..</td>
  <td style="text-align:right;">..kount_spending..</td>
  <td style="text-align:right;">..sumvat_spending..</td>
  <td style="text-align:right;">..sumrow_spending..</td>
</tr>
</table>
```

Below the code editor are three buttons: RUN >, Online form editor, Delete_Tables, and TABLES >. The TABLES button is highlighted.

The main area displays the generated table structure with columns labeled: Контрагент / Склад / Номенклатура / Ціна / Ціна, Кількість, Сума ПДВ, Вся сума, Кількість1, Сума ПДВ1, and Вся сума1. The table has two rows: one for general data and one for detailed breakdowns.

At the bottom of the interface, the text "Scr79" is visible.

Шапка таблиці буде виглядати так само, як у Підприємстві.

Якщо потрібно щось змінити — відкрийте рядок з тегом і відредактуйте його (наприклад, можна змінити ширину колонки через `width=?`).

Після цього знову натисніть кнопку TABLES, і зміни з’являться у структурі.

Кнопка Delete_Tables робить структуру таблиці невидимою.

У рядках з тегом також можна вносити зміни, але для цього потрібно добре розуміти, як працює HTML.

Головне правило: не змінюйте імена параметрів між двома крапками (наприклад, `..kount..`), інакше звіт перестане працювати.

На будь-яку колонку можна повісити подію.

Але обробку цієї події доведеться написати самостійно у підключенному модулі, використовуючи:

```
<script src="Myjsjs/METADATA __/id_n __/my.js"></script>
```

Якщо ви додасте нову подію, але програма не знайде відповідну функцію — це буде помилкою.

Таким чином, можна підключити власну обробку події для потрібної колонки.

У мене вже є приклад: на першій колонці вимірів стоїть подія

```
ondblclick="Dblclick_DOC(this)"
```

Вона за замовчуванням відкриває розшифровку по документах, які пов’язані з рядком.

Якщо вам ця розшифровка не потрібна — подію можна видалити або замінити на власну.

Тепер про кнопку **RUN>>**.

Вона дозволяє прямо у конструкторі звіту перевірити роботу запиту **SELECT....**

Якщо в запиті використовується відбір за датою (`dat BETWEEN dat8_ AND dat9_`), спочатку потрібно заповнити ці значення, інакше буде помилка.
Для цього натисніть кнопку **DATE** і у вікні, що з'явиться, вкажіть дати початку та кінця.
Після цього виділіть чистий текст запиту.

The screenshot shows the My4s application's query editor. The toolbar includes buttons for ADD_NEW, FROM, Return_VV_TAB, SampleTable, Same, and DATE. The DATE field is set to '01.01.2025' and '30.11.2025'. The main area contains a multi-line SQL query starting with 'WITH q_all AS ('. The code is a complex join between multiple tables (DOCS1, DOCS2, tab, vv) using various aliases like contr, store, nomencl, and price. It includes several GROUP BY clauses and UNION ALL operations. The entire query is enclosed in a large block comment block.

```
WITH q_all AS (
    SELECT Get_blob_A(vv,'contr') AS contr,Get_blob_A(vv,'store') AS store,Get_blob_A(tab,'nomencl') AS nomencl,Get_blob_T(tab,'price') AS price,SUM(Get_blob_A(tab,'kount')) AS kount,SUM(Get_blob_A(tab,'sumvat')) AS sumvat,SUM(Get_blob_A(tab,'sumrow')) AS sumrow,SUM(price_spending) AS price_spending,SUM(kount_spending) AS kount_spending,SUM(sumvat_spending) AS sumvat_spending
    FROM DOCS1 WHERE id_n='receipt' and dat BETWEEN dat8_ and dat9_ and 2=2
    GROUP BY contr,store,nomencl,price,price_spending
    UNION ALL
    SELECT Get_blob_A(vv,'contr') AS contr,Get_blob_A(vv,'store') AS store,Get_blob_A(tab,'nomencl') AS nomencl,NULL AS price,NULL AS kount,NULL AS sumvat,NULL AS sumrow,Get_blob_A(tab,'price') AS price_spending
    FROM DOCS2 WHERE id_n='spending' and dat BETWEEN 1735689600000 and 1764547199999 and 2=2
    GROUP BY contr,store,nomencl,price,price_spending
)
SELECT contr,store,nomencl,price,price_spending,SUM(kount) AS kount,SUM(sumvat) AS sumvat,SUM(sumrow) AS sumrow,SUM(kount_spending) AS kount_spending,SUM(sumvat_spending) AS sumvat_spending
FROM q_all
GROUP BY contr,store,nomencl,price,price_spending;
```

Scr80

От тоді лише можна натиснути кнопку «RUN>>».

Якщо з'явиться повідомлення «**undefined**», це означає, що з запитом щось не так!

Інакше у полі **textarea** з'явиться відповідь на запит приблизно в такому вигляді:

The screenshot shows the results of the executed query. The output is displayed in a large text area. It consists of several lines of text, each starting with a capital letter and followed by a value. These values correspond to the variables defined in the query: CONTR, STORE, NOMENCL, PRICE, PRICE_SPENDING, KOUNT, SUMVAT, SUMROW, and KOUNT_SPENDING. The values listed are: CONTR=BестТрейд//806, STORE=Головний склад//777, NOMENCL=Горячий Шоколад//802, PRICE=100, PRICE_SPENDING=null, KOUNT=300, SUMVAT=6000, SUMROW=36000, KOUNT_SPENDING=null, SUMVAT_SPENDING=null, SUMROW_SPENDING=null. Below this, there is a separator line consisting of many '+' characters. Following the separator, the output continues with CONTR=BестТрейд//806, STORE=Головний склад//777, NOMENCL=Горілка Козацька рада "Оригінальна" 0,75 л//790, PRICE=100, PRICE_SPENDING=null, KOUNT=210, SUMVAT=4200, SUMROW=25200, KOUNT_SPENDING=null.

```
CONTR=БестТрейд//806
STORE=Головний склад//777
NOMENCL=Горячий Шоколад//802
PRICE=100
PRICE_SPENDING=null
KOUNT=300
SUMVAT=6000
SUMROW=36000
KOUNT_SPENDING=null
SUMVAT_SPENDING=null
SUMROW_SPENDING=null
+++++
CONTR=БестТрейд//806
STORE=Головний склад//777
NOMENCL=Горілка Козацька рада "Оригінальна" 0,75 л//790
PRICE=100
PRICE_SPENDING=null
KOUNT=210
SUMVAT=4200
SUMROW=25200
KOUNT_SPENDING=null
```

Scr81

Не хвилюйтесь: при виведенні друкованої форми в «Підприємстві» усі дані програма обробить коректно.

Це означає одне — ваш запит працює.

Якщо кілька разів натиснути кнопку «вліво», то можна повернути поле **textarea** у вихідний стан, доки знову не з'являється значення дат у запиті (`dat BETWEEN dat8_ AND dat9_`).

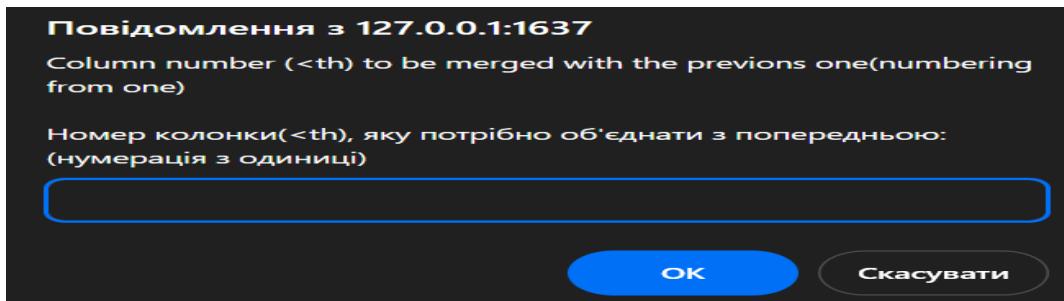
Перевірка кнопкою **RUN>>** не є обов'язковою. Досвідчені користувачі **My4s** і так зрозуміють, коли запит правильний.

А якщо він виявиться неправильним, у **Підприємстві** він просто не буде працювати — тоді ви повернетесь в Конструктор і вправите.

Тепер про кнопку **Same** (у тій же колонці).

Коли у звіті багато колонок, браузер намагається розмістити їх усі на екрані, і результат не завжди зручний.

Після натискання кнопки **Same** відкриється вікно, де потрібно вказати номер тега, який ви хочете об'єднати з попередньою колонкою.



Scr82

Автоматика об'єднає не лише у тегах `<th>`, але й у `<td>`.

Таким чином, натискаємо кнопку **Same** стільки разів, скільки потрібно об'єднати колонки — можна навіть у кілька рівнів!

Об'єднана колонка буде заремована.

Тут важливо, щоб у заповненні **Return** колонки йшли в такому порядку, щоб їх можна було правильно об'єднати.

Наприкінці можна зберегти всі зміни кнопкою «**SAVE METADATA**».

Потім можна перейти у **Підприємство**, вибрати в головному меню «**Additional reports (додаткові звіти)**», знайти свій звіт і перевірити його.

Якщо щось не так — повернутися у Конструктор, відредактувати звіт, знову його зберегти й перевірити у **Підприємстві**.

Звіти **query_fast** (Швидкі звіти)

Ці звіти багато в чому збігаються з попереднім звітом «**Помічник написання додаткових звітів**».

Для створення та редагування цих звітів не потрібен Конструктор — вони формуються безпосередньо в системі «**Підприємство**».

Така можливість дозволяє кожному користувачу (наприклад, бухгалтеру) самостійно створювати власні звіти.

Звісно, попередньо цьому повинні навчити програмісти.

Тому повторний опис не потрібен.

Відмінність полягає лише в тому, що у списку таблиць відсутні дві позиції: **BALANS_DOC** та **METADATA** (вважаю, що роботу з цими таблицями залишимо програмістам).

У «**Підприємстві**» в головному меню вибираємо пункт **query_fast** — відкриється форма з списком усіх швидких звітів.

© Update		METADATA: <input type="text" value="query_fast"/>	USER: <input type="text" value="demo"/>
© ADD NEW Query fast > >		© EDIT Query fast > >	
#	NAME	PRESENTATION	ACTUAL
1	CATS_CATS1	CATS_CATS1	✓
2	prov4	prov4	✓
3	prov5	prov5	✓
4	prov3	prov3	✓

Scr83

Якщо список містить велику кількість рядків, для пошуку потрібного можна скористатися стандартною комбінацією **Ctrl+F**.

Для запуску звіту виконайте подвійний клік по відповідному рядку.

У вже готових звітів немає обмежень за правами доступу.

Щоб створити новий звіт, натисніть кнопку **ADD NEW ...** — відкриється форма, аналогічна попередньому звіту.

Докладний опис можна знайти там же.

Для редагування існуючого звіту виділіть рядок і натисніть кнопку **EDIT**

Цими звітами можуть користуватися всі користувачі.

Однак редагувати їх можуть лише:

- користувачі (**user**), які створили звіт;
- користувачі з повними правами (**all**).

Важливо: якщо звіт редагує користувач з повними правами (**all**), основним власником все одно залишається той користувач, який його створив.

Processing (Обробки)

Знову ж таки за основу взято — **Помічник написання додаткових звітів.**



Практично завжди для обробки потрібен якийсь звіт. Це зручно ще й тому, що одразу можна перевірити коректність його виконання.

Якщо у фінальному варіанті вивід друкованої форми на екран виявиться зайвим, просто видаліть з **textarea** увесь вміст, починаючи з //TAB і нижче.

Можливий і інший сценарій: звіт при підготовці обробки взагалі не потрібен.

У цьому випадку всі процеси обробки ви описуєте самостійно в модулі **my.js** (або іншому вашому модулі, підключенному через **src**).

Тоді інформація в **textarea** не потрібна — її можна повністю видалити або навіть не починати заповнювати.

А в підключенному модулі:

обов'язково має бути функція: `function MyProcessing(zr) { // тут ви описуєте логіку обробки }`

Від вас уже залежатиме, як саме працюватиме обробка.

Підсумуємо звіти та обробки.

У системі «Підприємство» за роботу звітів та обробок відповідає єдиний HTML-шаблон **MyReport.htm** і підключенні модулі (окрім стандартних):

```
<script src="Report_db.js"></script>

<script src="Myjsjs/METADATA__id_n__/myXX.js"></script>

<script src="Mydbdb/query_fast.js"></script>

<script src="Myjsjs/METADATA__id_n__/my.js"></script>ort.htm
```

Докладний опис їхньої роботи я наводити не буду — код повністю відкритий.
Досвідчені програмісти зможуть без труднощів розібратися в логіці й за потреби адаптувати її під свої завдання.

Я лише зазначу кілька моментів, які допоможуть уникнути змін у модулі **Report_db.js**.
У більшості випадків коригування слід виконувати в модулі **my.js**, який спеціально призначений для цих цілей.

У **Report_db.js** передбачено кілька «реперних точок», які спрямовують виконання коду в модуль **my.js**. Саме через них зручно реалізовувати власні доробки, не порушуючи базову структуру.

```
if (typeof MyBeginRUN === "function") MyBeginRUN(strselect_); //My.js
```

Цей код перевіряє наявність функції **MyBeginRUN** у контексті виконання звіту чи обробки.
Без такої перевірки (якщо функція не буде знайдена) програма завершиться критичною помилкою.

Функцію **MyBeginRUN** можна використовувати для динамічної зміни рядка **strselect_** перед його виконанням у **Firebird**.

Якщо змін не потрібно — функція просто відсутня.

Наступна реперна функція:

```
if (typeof MyENDRUN === "function") MyENDRUN(zr); //My.js
```

Функція **MyENDRUN(zr)** дає можливість обробити вже отриманий результат з **Firebird** (якщо це вам раптом знадобиться).

Для обробок:

```
if (typeof MyProcessing === "function") MyProcessing(zr); //My.js
```

І ще одна:

```
if (typeof MyENDRUNrun === "function") MyENDRUNrun(tab_prn); //My.js
```

Тут можна спробувати змінити вигляд друкованої форми перед її виведенням на екран!

Firebird select

У Конструкторі є ще один звіт по метаданих — “**Firebird select**”.

Цей звіт я написав майже на самому початку створення **My4s**, коли у мене ще не було **Firebird Editor Pro**.

Він дозволяв мені швидко переглядати вміст таблиць **Firebird**.

А зараз я цей звіт залишив — можливо, комусь він буде цікавим.

The screenshot shows the Firebird Select interface. At the top, there are input fields for 'USER' (set to 'Программист'), 'WHERE:' (containing the query 'ID_N='operation' OR ID_N='moves''), and 'Order_by:' (set to 'id_n'). On the left, a sidebar displays a tree view of metadata tables like DOCS, ROWS_DOCS, METADATA, CATS, etc. The main area shows a grid of query results. The first few rows of the grid are:

100	moves P_M=false NUM=000000000001 DAT=1738772982123 COUNT_=	
	~@info=1~@userr=demo1~@accounts=8301~@accounts_=2821 Товари в роздрібній торгівлі (в ATT за продажною вартістю)1~@sto	
	00:00:00 UTC+0200 2025	
	moves P_M=false NUM=000000000002 DAT=1738862833467 COUNT_=	
	~@info=1~@userr=demo1~@accounts=8301~@accounts_=2821 Товари в роздрібній торгівлі (в ATT за продажною вартістю)1~@sto	
	00:00:00 UTC+0200 2025	
	operation P_M=false NUM=000000000001 DAT=1735848179710 COUNT_=	
	~@info=1~@userr=demo1~@infott=1~@infot=1~@datereg=2025-01-02T20:02:59.710Z1~@represent=Операція #000000000001 > 2025-	

Scr85

Описувати роботу звіту я не буду. Ті, хто знає **Firebird**, легко розберуться.

Робота з використанням різних часових зон

З'явилася можливість працювати клієнтам із базою, перебуваючи у різних часових зонах.

Клієнти можуть бути розташовані в різних країнах або в одній великій країні з кількома часовими зонами.

Як зберігати дати на сервері?

Робимо так, сервер, в якому б тимчасовій зоні не знаходився, завжди прив'язуємо до нульової зони Лондона (тільки зима!). А для кожного клієнта завжди обчислюємо зсув щодо Лондона (`timezoneOffset`) та при записі дат на сервері коригуємо дату клієнта на це усунення.

При отриманні звітів за датою знову ж таки використовуючи усунення, отримуємо звіт на тимчасову зону клієнта!

За промовчанням програма налаштована працювати в одній часовій зоні. У модулі `Blob.js`, на початку, є змінна: «`let timezoneOffset = 0;`». Якщо цій змінній надати значення 1, тоді і почне працювати програма з різними зонами!

Увага! Не можна цю константу змінювати у процесі роботи. Вона повинна налаштовуватися на самому початку впровадження My4s!!

Підсумуємо вище написане.

Я добре розумію, що викладене місцями може бути не зовсім зрозумілим одразу. Але цього опису цілком достатньо, щоб ви вирішили — хочете ви продовжити ознайомлення з **My4s** чи ні.

Якщо хочете продовжити, тоді спершу потрібно встановити демо-версію й пройтися по опису вже на конкретних прикладах.

Тоді багато речей стане зрозумілішими.

Керівництво по встановленню My4s на ваш комп'ютер або ноутбук з Windows

У вас на комп'ютері вже повинен бути архів з дистрибутивом **Distrib_My4s.zip** перенесений з **GitHub**!

Увага! Демо-база і порожня база (my4sNull) у мене працюють з кодуванням WIN1251.

Я створював **My4s** у першу чергу для своєї країни — України.

Це не означає, що з іншим кодуванням система не буде працювати, але для її налаштування доведеться трохи постаратися (штучний інтелект вам на допомогу!).

Якщо у вас встановлена **Windows 10** або вище (на **Windows 8.1** теж має працювати, але це не перевірялося), подальші кроки виконуються без проблем.

Створення папок і бази

- Створіть на комп'ютері папку для зберігання баз **Firebird** (наприклад, *FB*).
- Усередині неї створіть папку для демо-бази (наприклад, *demoFB*).
- Перенесіть файл **MYDB9.FDB** і **MYDB3.fbk** з дистрибутива в цю папку.
- Скопіюйте файл **fbclient.dll** з папки Firebird у папку з демо-базою.

Копіювання додаткових файлів

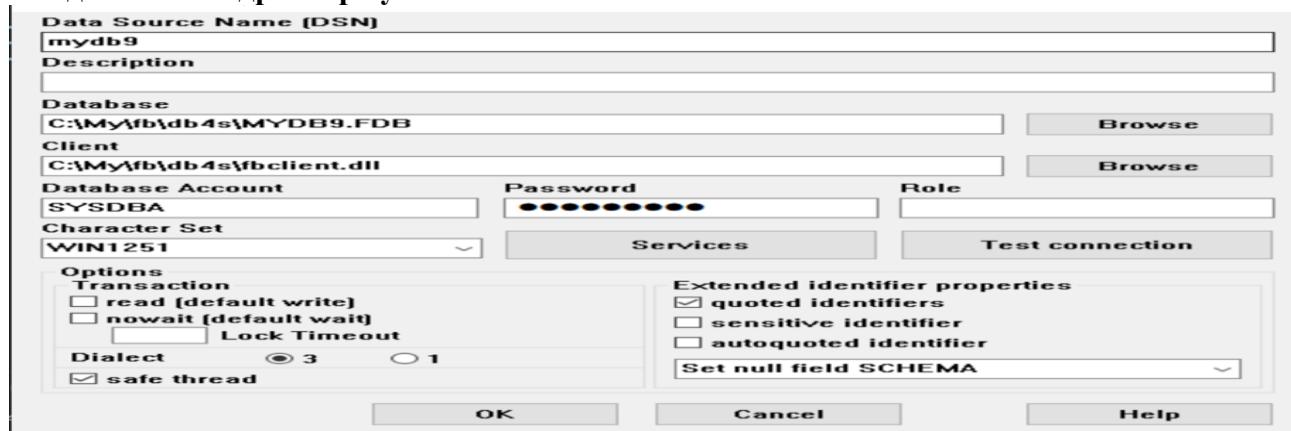
Із дистрибутива скопіюйте папку **My4s** у зручне вам місце на комп'ютері.

Встановлення Firebird

- Встановіть стабільну версію **Firebird 3.x** (для ознайомлення цього достатньо; вона легша за розміром).
- Можна використовувати версії **4.x** або **5.x**, але для початкового ознайомлення рекомендується **3.x**.

Встановлення ODBC драйвера

- Завантажте драйвер **Firebird_ODBC_2.0.5.156_x64.exe** (або більш свіжу версію, якщо вона доситьна).
 - Встановіть драйвер.
1. Підключення драйвера у Windows



Scr86

Заповніть як на скрині.

Для роботи з демо-базою обов'язково DSN має бути **mydb9** та **Character Set WIN1251** (у мене це вже прописано в налаштуваннях).

Далі правильно заповніть **database** і **client**.

Пароль — **masterkey**, а все інше як на картинці.

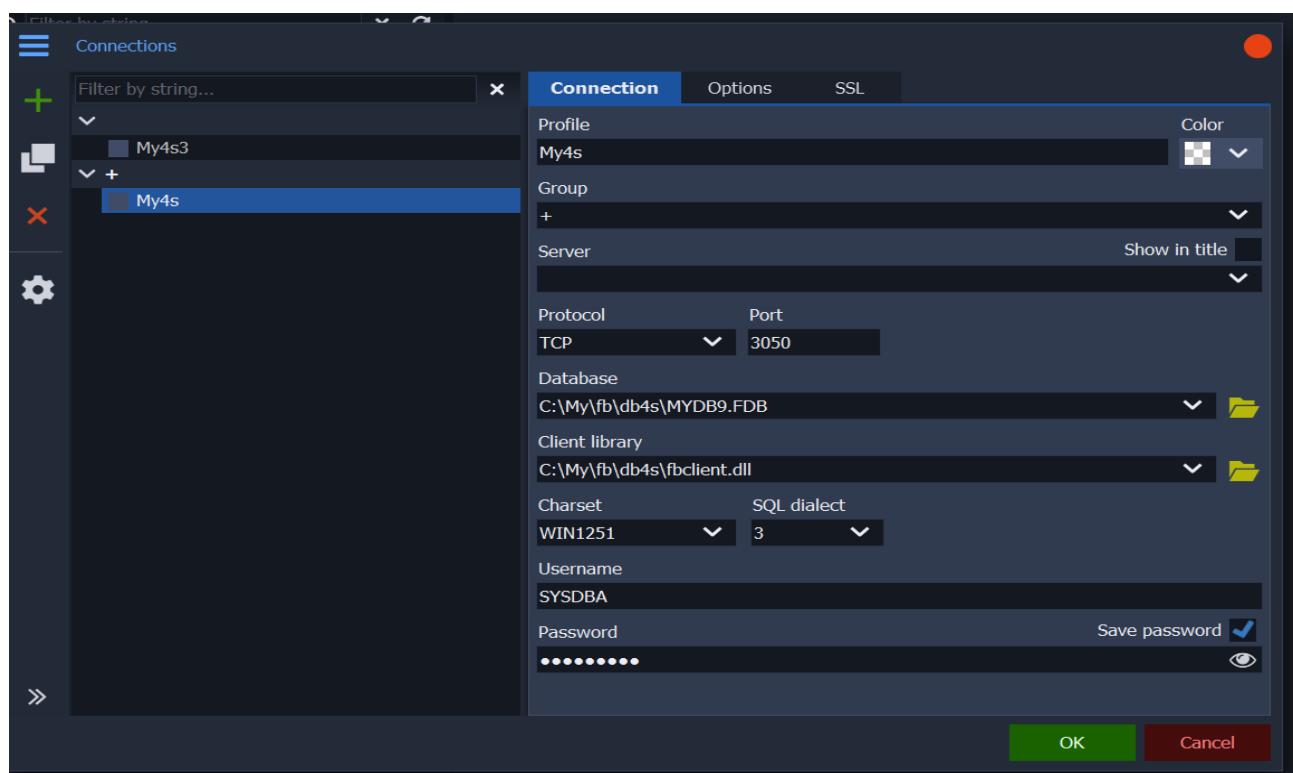
Натисніть **Test connection** — якщо помилки не буде, тоді **OK**.

Встановіть Firebird Editor Pro.

Якщо у вас вже є інший редактор, тоді ви, мабуть, уже досвідчений користувач щонайменше Firebird і розбереться самостійно!

Відкриваєте «**Firebird Editor Pro**».

Натискаєте плюсик (**New connection**) і заповнюєте форму:



Scr87

У мене це **My4s**, але ви можете назвати по-своєму, наприклад **demo4s**.

У полі **Database** виберіть файл з демо-базою, а у полі **Client library** — файл **fbclient.dll**.

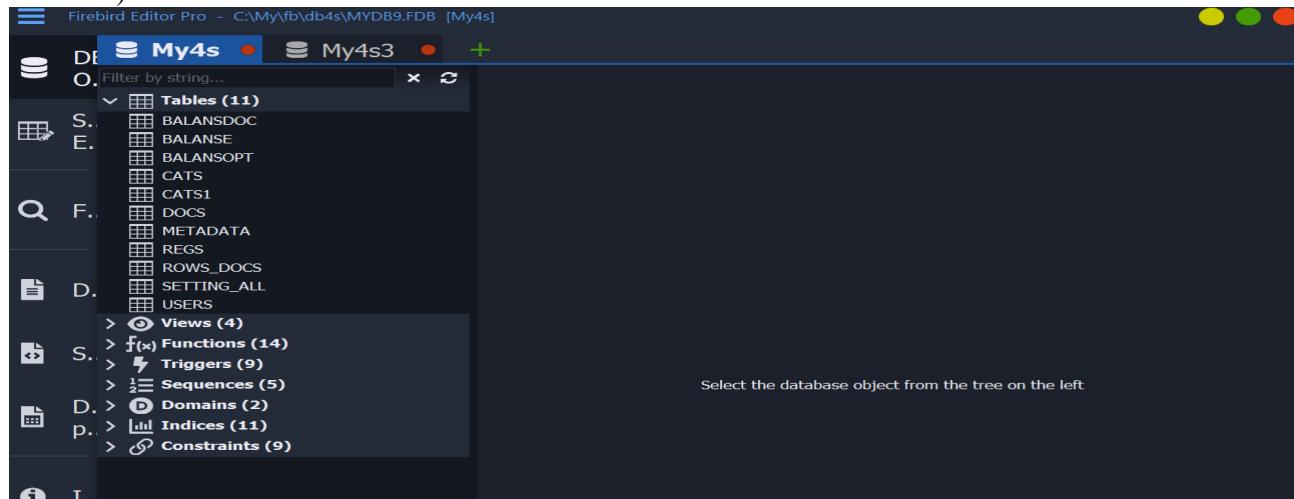
Увага! При створенні Бекапів і Ресторе обовязково перевибирайте ці файли, навіть коли вони показують правильний шлях!

Для демо-бази у мене використовується стандартний пароль — **masterkey**.

Все інше має бути як на картинці!

Потім натискаєте **OK**.

Якщо заповнили все правильно, то у вас з'явиться демо-база (у мене це **My4s**, а у вас буде так, як ви її назвали).



Scr88

Далі встановіть **Node.js** — останню стабільну версію (LTS).

Якщо вам байдуже, можна обрати встановлення за замовчуванням.

Після цього (якщо ще не встановлено) встановіть стабільну версію **Visual Studio Code**.

Запустіть редактор і відкрийте папку (вона вже має бути скопійована з дистрибутива) **My4s** через меню **File → Open Folder**.

Зліва з'явиться колонка провідника з усім вмістом папки **My4s**.

Ви зможете відкривати будь-які файли, переглядати їхній зміст і за потреби вносити зміни.

Одним кліком оновлювати файли й одразу в браузері дивитися, що ви змінили (при цьому не потрібно знову заходити в **My4s** з нуля, достатньо лише перезавантажити ту сторінку, яка стосується ваших змін!).

Visual Studio Code — це потужний текстовий редактор,

у нашому випадку його головна перевага полягає в хорошій підтримці **Node.js** та можливості відлагодження модулів. . Можна запускати в режимі відладки!

Крім того, редактор підтримує різні мови інтерфейсу, що робить роботу більш зручною.
Опис ви знайдете в інтернеті.

🔧 Найпопулярніші розширення для Node.js у VS Code

- **ESLint** — допомагає підтримувати єдиний стиль коду та знаходить помилки ще до запуску.
- **Prettier – Code formatter** — автоматично вирівнює та форматує код, щоб він виглядав охайню й однаково у всіх файлах.
- **Debugger for Chrome** — дозволяє відлагоджувати **Node.js** та фронтенд-код безпосередньо з **VS Code**.
- **npm Intellisense** — підказує імпортовані пакети прт, щоб швидше писати код.
- **Path Intellisense** — автодоповнення шляхів до файлів і папок у проекті.
- **REST Client** — зручно тестувати API-запити прямо з редактора.

Ці розширення значно спрощують роботу з **Node.js**, особливо якщо ви тільки починаєте.

Робота з My4s під Windows 7

Якщо на вашому комп’ютері встановлено **Windows 7** — не хвилюйтесь.
У дистрибутиві **Windows7** ви знайдете:

- **Node.js версії 12** (сумісний з Windows 7)
- Папку з готовим **VS Code**, який також працюватиме під Windows 7

Кроки для налаштування бази

1. Для ознайомлення з **My4s** достатньо демо-версії.
2. Якщо ви хочете створити власну базу з нуля:
 - У дистрибутиві є папка **my4sNull**, що містить:
 - початкову базу **mydb3.FDB**
 - резервну копію **mydb3.fbk**
 - У папці **FB** на вашому комп’ютері (або в іншій папці з будь-якою назвою) створіть нову папку **MyFb**.
 - Перенесіть туди файли з дистрибутива **my4sNull**.
3. У папці **My4s** вже є заготовка під початкову базу в папці **mydb3**.
4. У налаштуваннях Windows додайте новий **DSN mydb3** для ODBC-драйвера.
5. У **Firebird Editor Pro** створіть нове підключення (наприклад, **My4s3 або по іншому**, як описано вище).

Перевірка роботи

- Якщо все зроблено правильно, ви зможете працювати одразу з двома базами у браузері:
 - **Демо-база:** `127.0.0.1:1637/mydb/cf` або `127.0.0.1:1637/mydb/db`
 - **Ваша база:** `127.0.0.1:1637/mydb3/cf` або `127.0.0.1:1637/mydb/db`
- Пароль за замовчуванням — **masterkey**. Його можна змінити на будь-який інший.
-
- **!** Якщо з першого разу не вийде — не засмучуйтесь. Це нормально, в мене теж не одразу вийшло! Просто уважно перечитуйте опис.
- Для програмування «з нуля» або роботи з демо-базою на локальному комп’ютері цілком достатньо використовувати **VS Code** — адже там можна безпосередньо запускати **Node.exe** через меню **Run (Виконати)**.
Але що робити, коли ви вже створили свою програму, відлагодили її і розмістили на сервері папку **My4s**, щоб з базою могли працювати багато користувачів через інтернет або локальну мережу?

Для цього на сервері, де встановлено **Node.exe** (нагадаємо: у My4s сервером виступає лише один файл **Node.exe**, який має знаходитися в папці **My4s**), передбачено спеціальний bat-файл **Start_Node.bat**. У його вмісті всього один рядок:

node.exe Start4s.js

При запуску **Node.exe** отримує на вхід файл **Start4s.js**, який:

- слухає порт (у даному випадку **1637**),
- містить основну маршрутизацію,
- підключає решту файлів з папки **My4s** у міру необхідності.
- **Достатньо запустити цей bat-файл — і ви зможете підключатися до бази прямо з браузера!**
Важливо: на сервері необов’язково встановлювати Node.js повністю. Головне, щоб у вас був потрібний файл **Node.exe** саме того релізу, який потрібен для роботи!
Увага! Не можна одночасно запускати **Node** з **VS Code** і bat-файл, якщо ви працюєте на одному комп’ютері!

Зв'язок з автором:

Адреса електронної пошти для контакту: sergey2429@i.ua

Буду радий, якщо знайдете якісь неточності, баги, не дай Боже помилки. Обов'язково опишіть, за яких умов це виникає!

Якщо самостійно доопрацюєте якусь частину програми — детально опишіть, що саме ви зробили. Якщо я вважатиму, що це справді покращує програму, з задоволенням внесу зміни до програми та зазначу вас як автора цієї частини!

Допомога проекту:

Картка АТ "Ощадбанк" в грн. **5167 8032 1042 4912**

Ліцензія та використання(MIT)

My4S можна використовувати безкоштовно в будь-якому середовищі, включаючи, але не обмежуючись: особистим, академічним, комерційним, державним, корпоративним, некомерційним та комерційним застосуванням.

"Безкоштовно" означає відсутність плати за встановлення та використання My4S. Будь-яка людина, яка отримала копію цього програмного забезпечення, має право використовувати його без обмежень, включаючи: використання, копіювання, модифікацію, публікацію, розповсюдження, надання доступу іншим користувачам з аналогічними правами.

Що означає ця ліцензія

- **Безкоштовність:** ви не платите ні за встановлення, ні за використання My4S.
- **Свобода дій:** можна запускати програму в будь-якому середовищі — вдома, на роботі, в університеті чи навіть у державних установах.
- **Повні права користувача:** дозволено копіювати, змінювати, поширювати програму та навіть ділитися нею з іншими.
- **Відсутність обмежень:** немає прихованих умов чи ліцензійних платежів — усе відкрито й прозоро.

Авторські права

На всіх програмах, створених за допомогою My4S, необхідно зазначати, що вони розроблені з використанням цього програмного забезпечення.

! Пояснення для новачків: це правило означає, що якщо ви зробили власну програму, використовуючи My4S, потрібно вказати джерело — тобто написати, що вона створена за допомогою My4S. Це своєрідна форма визнання та поваги до інструменту, який допоміг вам у розробці. Така практика поширена у світі відкритого ПЗ і допомагає іншим зрозуміти, які технології були використані.

Відмова від гарантій

Програмне забезпечення надається "ЯК Є", без будь-яких гарантій, явних чи неявних, включаючи, але не обмежуючись: гарантіями комерційної придатності, відповідності певній меті, ненарушення сторонніх прав.

Автори та правовласники не несуть відповідальності за будь-які претензії, збитки чи інші зобов'язання, що виникають у результаті використання цього програмного за-безпечення.

