AAM projects commints

11/15/2023

Fixing, refactoring, fifo triggers and PL

Allocation Trades UI : Bulk order and trade details linked to allocated trade via JSON&Dialog

DB - FIFO Triggers: Fifo triggers to for creation updated. Added restriction to add back traded transaction in order to ensure fifo sequence ;

DB && UI Creating acconting: Notification if there is no root depo account for given portfolio

Fixed currency validator => When currency list is obtained, all subscribers are notified and they trigger currency fileds validation

Trade form UI fixed issue with opening allocation tab and correctly requesting orders list for allocation adding status array for query

Reduced queries to get first opened Accounting Day

Entry Form UI: Fixed overdraft validator for entry created based on existed one. There was no need to exclude id of orginal entry from calculation

11/16/2023

Portfolio Tabs UI=> Created dynamic tab label. Portfolio code is included in the active tab label

Tree menu UI => slightly changed design and blocks Trades&Orders, Accounting have been exluded from favorite as hardore fix inside intiation function

Trade Table UI => default filter has been changed to multiFilter

Entries, Orders, Allocation Table UI => default filter has been changed to multiFilter

Orders Table => Fixed reloading and onChange behavior, taking into account table mode and the parent of the view

Entries Table UI => Blocked UI to modify trades entries linked to FIFO calculation

--------------------------------------------------------------------------------------------

11/17-18/2023

1|| Orders Creation DB function and UI => Blocked creation of new orders with oppositie direction to the exist ones. For examlple if for any given pair portfolio - secid of potential buy order there is a sell not-accounted order then DB function f\_i\_o\_create\_orders\_by\_mp will generate an error message. The main issue with implementing is to generate exception in postgres function and then use the same dataset one more time to insert new orders. Using SELECT INTO in WITH clause is forbidden and to avoid duplicate query the decesion is to create temp table to store portential new orders

2|| Orders Table UI = > added status filter for retrived dataset and status filter for complex search

11/20/2023

1|| Entry Form UI: => Fixed unavaible data about expected balance in disabled form, view mode. Disable instruction was moved to ngAfterInt lifecycle hook. thus all form data has been initiated before disabling controls elements

2|| Balance sheet UI:=> planned trunovers => Fixed DB function f\_a\_b\_bcurrent\_ledger\_turnovers\_balances\_notclosed by adding missing '\*' to properly calculate negative transactions

3|| Orders-Trades UI: => Option to book a new trade based on order data was added to orders table. Fixed creating bond trade. It was required to pass parameter to the trade form to run secidChange functions to add data regarding face value, faceunit and etc in order to calculate trade amount correctly

4\_\_\_ Creating testing data to calculate time weighted rate of return

--------------------------------------------------------------------------------------------

11/21/2023

1|| Portfolios NPV historical data to calculate time weighed roi and build chart of portfolio value dynamic change => DB function created => Using balances dynamic data from function f\_a\_b\_balancesheet\_total\_closed\_and\_notclosed() for given period we calculate portfolio npv for every date. Balances data from last closed acconting date (or from the start of the period if its closed date) to the end period copied to the temp table and is used to calculate portfolio positions (using lateral join) for every date from the period. Then using lateral join

we get closest market price for every position

2|| Market Chart UI => restored sorting order for retrived market data to build chart correctly

3|| Allocation Table (Portfolio Trades Tab) => For portfolio trades tab modying functions are closed like creating/deleting accounting or deletion of trade

11/22/23

1|| TW roi

2|| Opening date for new depo accounts. Fist opened accounting date is used as opening date for new depo accounts

3|| Balances closed till november and reconciled. Difference rounded to 2 digts

4|| UI Accounts form => added date opeining for modying and viewing

5|| Opening date updated according to the first entry transaction date by the given account

11/23/23

1|| AutoCompleteService created functionality to notify components about arriving of secid autocomplete list

2|| Account Form UI => Added updating of secid validity on receiving notification that secidList is ready

3|| Account Form UI => Implemented control element to modify date opening of account and modify backend accordingly

4|| Martket Prices DB function refactored (using lateral join and etc)=> As a result portfolio structure function was speeded up by 40%

11/24/23

1|| DB f\_i\_get\_npv\_dynamic\_with\_performance\_data.sql => refactoring

2|| DB created function f\_i\_get\_accured\_interest\_for\_period\_secidlist() to fetch coupon periods for every date with report period to calculate npv

3|| DB created function f\_i\_get\_cross\_ratesfor\_period\_currencylist() to fetch cross rate for every date with report period to calculate npv

--------------------------------------------------------------------------------------------

11/25-26/23

1|| DB Fixed functions: f\_i\_portfolios\_balances\_by\_account\_for\_idportfolios.sql,f\_i\_get\_accured\_interests\_for\_portfolios.sql,f\_i\_get\_cross\_ratesfor\_period\_currencylist.sql,f\_i\_get\_npv\_dynamic\_with\_performance\_data.sql

11/27/23

1|| DB Fix f\_i\_get\_accrued\_interest\_for\_period\_second list and f\_i\_get\_accrued\_interests\_for\_portfolios. Coupon periods have been corrected. Report date have to be greater than start date since start\_date equal to the end date of the previous period

11/28-30/23

1|| Creating compound correction rate to calculate time weighted return of investment taking into account all deposits and withdrawals during investment period

2|| DB v2f\_i\_get\_npv\_dynamic\_with\_performance\_trio refactoring, debugging and reconcile data

12/1/23

1 || DB refactored v2f\_i\_get\_npv\_dynamic\_with\_performance\_trio and as a result execution time for maximum load dropped from 7 sec to 1 sec

2 || Tested and Verified performance data generated by v2f\_i\_get\_npv\_dynamic\_with\_performance\_two function

12/2/23

1|| Reconcile performance data

2|| UI Trade form => developed DB function f\_i\_get\_cross\_rate\_for\_trade to retrive current cross rate based on user choice of base ,quote and set by dafault cross currency. Rate is used as an indicator to reduce risk of manul error. Indirect rate calculated as 1/rate has been added in hint section of settlement rate filed to facilitate reconcile

3|| UI Currencies Table => Indirect rate has been inclueded in the table

3/12/23

1|| UI Trade form => indicative market price info and confirmation of diffrence more than 20%. Created DB function to retrive market price f\_i\_get\_market\_quote\_for\_trade.sql

2|| Fixed trades with testing trades with incorrect settlement and price Currencies

3|| Closed all balances till October

--------------------------------------------------------------------------------------------

4/12/23

1|| DB refactoring cross\_rate functions and implementing option to evaluate dynamic portfolio perfomance in different currencies 978,826,156,810,756

2|| UI Account details form => refactored accounts date of opening validation based on min date as firt opened accounting day. Validation uses current date which could be in closed period until it modifed. After that event min validation moves to first opened accounting day

5/12/23

1|| DB function f\_i\_get\_cross\_ratesfor\_period\_currencylist has been finished and is being used to get cross rates set for report period

2|| DB updated functions PSQL\_Quaries/f\_i\_get\_npv\_dynamic.sql and PSQL\_Quaries/f\_i\_get\_portfolios\_structure\_detailed\_data.sql to retrieve only required cross currency rates and apply them correctly

3|| UI Table to track portfolios npv dynamic perfomance, roi = > editing draft of the table

6/12/23

1|| UI Table to track portfolios npv dynamic perfomance, roi = > fully implemented with complex search, dynamic currency symbol change, excel export and formatted data

2|| UI Chart Performance analysis => Time-Weighted Rate of Return => draft of the chart

--------------------------------------------------------------------------------------------

7/12/23

1|| UI Chart Performance analysis => Time-Weighted Rate of Return => draft of the chart

10-11/12/23

1|| UI Chart Performance analysis implemented:

Custom color model for cash outs/in,

Dynamic labels and tooltips populating based type of data and type of transfer (deposit or withdrawal)

Custom/dynamic yAxises: design depends on data series and current scale

Added MarkPoints for TWR line to show max,min and the latest value of perfomance rate

2|| UI NPV Dynamic Table => create draft

13/12/23

1|| UI NPV Dynamic Table => Implemented table with complex search and design patterns

2|| DB & UI Profit and Loss calculation =>

Fixed costs calculation for local bonds

Implemented costs and pl conversion according to the report currency value

--------------------------------------------------------------------------------------------

14/12/23

1|| UI Portfolio position => User allowed to retrieve position data passing date and report currency as parameters. The algoritm has been changed by replacing pattern of storing the whole dataset and filtering it accordingly with quering only nessesary live data

--------------------------------------------------------------------------------------------

15/12/23

1|| UI Portfolio details Tab => Added performance chart rendering for chosen portfolio onChange

2|| UI Clients details Tab => Added performance chart rendering for the first portfolio of the given client. User can chose other portfolios from the select list. Data for them has been preloaded

3|| UI Strategy/Model Portfolio details Tab => Added performance chart without any preloaded data. The goal is to provide investment manager with all needed tools in one tabGroup

--------------------------------------------------------------------------------------------

16/12/23 - 18/12/23

1|| DB&Model PL detail analysis => Created DB function f\_i\_get\_pl\_dynamic\_with\_npv.sql to provide data for detailed factor analysis of clients revenue. Function calculates realized profit and loss based on fifo data and unrealized profit/loss comparing current position evaluated by market price and costs of creating position retrieved from fifo data. An idea is to provide an UI to create chart showing the dynamic of portfolio revenue splitting it by separate instruments and graphicly demonstrating its contribution in final financial result of managment

19-20/12/23

1|| UI Performance Chart =>

Implemented stack area chart for revenue structure analysis in dynamic.

Chart:

Series are created dynamicly based on portfolio instrument list.

DataSets are aligned according to xAxis date series by adding nessesary array of zeros value before the actual data.

Legend items are sorted according to the order of areas in the chart

Created UI and API to fetch dataSet to draw a chart.

Realized complex filters data retrieve and export to excel for further manual analysis and etc

--------------------------------------------------------------------------------------------

21/12/2023

1|| UI Performance Chart Factor Revenue =>

Fixed aligning order of labels according to the logic of chart engine. Thus the chart is more readable. Labels take the place according to the time of appearing first income or loss and the final type of revenue: profit or loss as well

Made a custom tooltip with table splitting total pl by separate instrument contributions and added custom style

Chart has been added to portfolio's tab group. By default it demonstrates revenue factor analysis for the last three months and could be customize by the user if it's nessesary

22/12/2023 - 24/12/2023

1|| Managment Fees Backend: DB model and DB functions =>

Created db tables for fees calculation and processing

Created db functions to calculate managment fees and save calculation for futher processing

2|| UI Managment Fees => Created base UI draft table

--------------------------------------------------------------------------------------------

25/12/2023 - 26/12/2023

1|| UI Management Fees =>

UI splitted into two tables calculation and processing

Added default functionality: complex search and excel. excel export has been refactored

Created function to save prepared calculation in relevant table for further proccessing. User is able to see which calculations have been saved earilier and is notified only new calculation will be saved.

Created UI function to show detailed data of calculations or grouped by portfolio

Created delete stored calculations function for selected items.

DB trigger on delete created to verify because user is only allowed to delete calculations without accounting

--------------------------------------------------------------------------------------------

27/12/2023 - 28/12/2023..

1|| UI&Backend Fees proccessing =>

Added accounting scheme to process fees accounting.

Accounting scheme consists of three entries. A fee amount is transferred to transit account and then distribute it between net profit and taxes due to be paid

Implemented basic functionality to store profit tax rate and use it for taxes calculation

Developed function to create accounting related to the fees transaction using accounting schemes stored in DB and which could be customized if it is nessesary

Within accounting creation function reference created between fees transactions and entry corresponding to the client account

Used forkjoin operator and other RxJS functions

2|| UI Confirm dialog has been refactored

3|| Minor fixing

--------------------------------------------------------------------------------------------

29/12/2023 - 30/12/2023

1|| UI&Backend Fees proccessing =>

Changed type of reference between accounting and fee calculation transaction. New reference contains array of entries assosiated with the calculation.

Developed DB function to delete accounting linked to the calculations and updating corresponding fees calculations

Add UI to delete acconting for fees calculations and calculations

2|| Minor fixing (entries list complex search by entry type fixed)

--------------------------------------------------------------------------------------------

1/1/2024 - 3/12/024

1|| UI Fees processing => Added function to show list of created entries linked to the saved fee calculation

2|| UI entries list => Implemented option to retrieve list of entries based on array of entries ids passed as a parameter

DB function edited by adding new parameter and changing query conditions respectively

Node JS changed logic to treat array parameters

3|| UI&Backend Fees proccessing => Implemented option to set acconting date for fee entry transactions

Added datepicker with minDate restriction as MIN(firstOpenedAccountingDate||MAX(saved fees calculations in the table)). Thus the accounting could be created not earilier than the last date in the calculation period

Accounting date information has been added in DB functions which are responsible for updating/removing reference between fee calculation row and entries generated

4|| UI entry form fixing => Fixed overdraft overriding status

--------------------------------------------------------------------------------------------

5/12/2024 - 8/1/2024

1|| UI & DB Entries Table =>

Added function to delete bulk of entries

Refactored UI design and ts

Refactored logic of allowing manual editing of acconting entry => Each entry group has boolean switch (DB table) which can exclude auto-generated entries from manual corrections to ensure data consistency

2|| UI & DB Fees proccessing table => Current account balance and calculated balance after fees deduction have been added to the UI table

DB: new function is created to calculate current account balances based on portfolio codes and date as parameters.

UI has been updated. Account balances upload on two events: fees dataset has been reloaded or acccounting date has been changed

--------------------------------------------------------------------------------------------

9/1/2024 - 10/1/2024

1|| UI Fees proccessing table =>

User notification about potential overdraft after fees deduction has been added to the accounting creation pipe. User is able to confirm overdrafts or cancel entries generation.

2|| Performance Fees =>

Created calculation model in excel

DB functions created:

f\_f\_calc\_performance\_fees - To calculate performance fees for given portfolios list and for given period

f\_f\_get\_performance\_fees\_schedules - To get performance fees schedules, rates, hwm and other data needed for calculation

f\_i\_get\_deposits\_withdrawals\_per\_portfolios\_on\_date - Function to get all non trades related cash movements to compute PnL

3|| Pnl Data reconcilation =>

corrected pnl data on 16 sep 2023. Incorrect data was left after fifo implementation

changed type of early testing transactions to deposit and withdrawals. transactions were left after swift stp implementation

--------------------------------------------------------------------------------------------

11/1/2024 - 13/12/2024

1|| DB&UI Deducted fees (management, performance) are included in UI portfolio positions detailed data table.

Amount of fees is taken into account when PnL is calculated.

Created Db function f\_i\_get\_deducted\_fees\_per\_portfolios\_on\_date in order to obtain amounts of fees deducted.

New function has been added to middleware layer of getting portfolios data to show client statament

2|| High-Water Mark perfomance fee =>

High Water Mark(HWM) option has bee added to DB function f\_f\_calc\_performance\_fees, which calculates perfomance fee

HWM is saved for every calculation and it is changed when PnL reaches new max. New calculation gets the latest HWM and evaluates PnL above it as a subject to fee deduction

HWM has been added to fee schedules function to calculate new fees based on the latest HWM

3|| Performance Fee UI & DB =>

Created DB function to insert new calculations in the dfees\_transactions table

Created separate table component to calculate perfomance fees

Created separate table component to process perfomance fees

4|| Performance Fees Accounting => Created accounting schemes to process perfomance fees

5|| UI & ML Ledger Accounts Form => exposed dateOpening field for editing. All relevant validation has been applied

--------------------------------------------------------------------------------------------

14/1/2024 - 15/1/2024

1|| Done: Check managemnt fee calculation when rate has been changed during period

2|| \* Fix - Null NPV for current date from f\_npv\_dynamic

Fixed - historical period for exchange rates has been increased upto 60 days

The better way is to upload exchange rate more frequently

3|| Done: Check perfomance fee calculation when rate has been changed during period.

Best and correct approach is to 'close' performance fee period by calculating an amount of fee on the last date of the period. Fee amount without hurdle rate doesn't depend on the period length

4|| Fixing =>

Performance and Management fee corrected removing saved calculations and while processing removing fees with accounting

Performance Fees based on entries type have been added to detailed position table

5|| DB creating performance fee transactions =>

Created trigger which prevent creating new calculation before the date of the last calculation

6|| Hrudle rate implementation => Draft

--------------------------------------------------------------------------------------------

16/01/2024 - 19/01/2024

1|| UI & DB Fees Dictionary Table =>

Created table of fees with child tabele of schedules

2|| Created service to notify child schedule table that it was opened and to retrieve data if its nessesary. Thus schedules are requested from DB only if user is interested in them and DB workload has been Reduced

3|| AutoCompleteService - ModelPortfolios Pipe =>

ModelPortfolios pipe with exhaustMap has been added in order to avoid multiple request to DB for the same ModelPortfolios autocomplete list

Service functions using subject-observable pattern have been created to provide subscribers with requested list when it is ready

4|| UI & DB Forms for FeeMainData and FeeSchedulesData =>

Two forms have been created to provide UI for updating fees dictionary data (schedules and main data)

Middleware functions and services have been added to modify fees data

In order to avoid unnecessary workload to DB data returned modified data have been used to update Tables DataSource respectivly

5|| Fixing

--------------------------------------------------------------------------------------------

20/01/2024 - 23/01/2024

1|| UI & DB Portfolios and linked fees schedules Table =>

Interface uses portfolio table and show as child table fees schedules linked to the selected portfolio. Child fees schedule table has its own child table which allows to get info about fee schedules ranges and etc

2|| UI & DB Form to assign fees schedules to client portfolios and modify created data =>

Form allows to create/update/delete links between client portfolio and fee schedules. Form shows fee schedules ranges for the selected fee in readonly mode as reference info

3|| Minor file names fixing

--------------------------------------------------------------------------------------------

24/01/2024 - 25/01/2024

1|| DB Validation Fees Schedules ranges insert|update=>

Created DB trigger to prevent schedules ranges overlapping existing ones

2|| DB Validation linking between client portfolios and fees schedules insert|update=>

Created DB trigger to prevent links with overlapping applying periods for given portfolio and fee type (Performance/Managment)

3|| DB Validation preventing fees main data modifying if there are created calculations based on it =>

Created DB trigger to prevent updating dfees\_main if there are rows with fee's id in the dfees\_transactions

4|| DB Validation preventing fees schedules ranges data modifying if there are created calculations based on it =>

DB trigger on dfees\_schedules has been updated to verify there are no rows with main fee's id in the dfees\_transactions

5|| Fee calculations deletion fixed

--------------------------------------------------------------------------------------------

26/01/2024 - 28/01/2024

1|| DB Validation preventing link portfolio <=> fees schedule modifying if there are calculations created based on it=>

DB trigger on the table dfees\_objects has been updated to allow modify only period\_end field and on condition old or new value not less than last calculation saved in table dfees\_transactions

2|| UI Fee Dictonary Table => Added function to show a list of portfolios attached to the selected fee schedule

3|| Performance Fee DB => Fixed multiple ranges calculations for performance fee => One range is taken based on amount of profit exeeding high water mark value

--------------------------------------------------------------------------------------------

29/01/2024 - 3/02/2024

29/01/2024 - 3/02/2024

1|| UI&DB FIFO detailed reporting =>

Created DB function f\_fifo\_select\_all\_trades to select nessesary fifo transactions. The function accepts price range, qty range, date range, lists of portfolios, secids, trades ids as filter parameters. Used new pattern when filter is ignored when its value is null and are used ranges in where clause. Added index based on order by statament

Created UI table to show FIFO transactions with option to request data using complex search with the same parameters as in DB function. Used class condition on rows to apply different style to close and open transactions.

Refactored using of ranges as search parameters

Refactored using of chiplists as search parameters. Got rid of additional array and reduce functions to the chiplists. HTML template were edited accordingly

2|| UI Portfolio Form =>

Added table with fees linked to the portfolio with option to edit them

Summary Tab received data regarding current PnL, NPV and Fees via service and displays it within the form when it's opened via menu tree

3|| Common form to render JSON result of db request has been refactored =>

Used custom json transform procedure to populate form and added function to export result in excel format. Changed from style in HTML template

4|| Middleware server logging => slightly changed logging data message

5|| Middleware common functions => function to trasfrom arrays according to its type for get requests (when numbers arrives as strings) was moved to shared module

--------------------------------------------------------------------------------------------

4/02/2024 - 5/02/2024

1|| FIFO positions (portfolios) on date reporting =>

Created DB function f\_fifo\_get\_cost\_pl\_detailed and summary.sql to select nessesary fifo positions. The function accepts report date (if omitted by default current date is taken), list of portfolios, list of secids as filter parameters. The function provides with trades details remained in the fifo position on the requested date and demonstrates aggregated position data as well

Created UI table to show FIFO positions(aggregated position and trades included in it) with option to request data using complex search with the same parameters as in DB function.

--------------------------------------------------------------------------------------------

6/02/2024-8/02/2024

1|| Store histrory of a strategies changing structure DB =>

Created new DB table public.dstrategies\_global\_structure\_history to store historical data of model portfolio structure changing. Table is a replica of the main table with additional fields to keep track of changing (user, time, type).

Implemented new trigger trg\_f\_i\_dstrategies\_global\_structure\_history.sql to create historical transactions when data in the main table has been modified. Every update transaction creates to two transactions in the history table: one contains new data and the other old data (before update). Delete and Create make only one transaction with the respective type

Added new DB function f\_i\_model\_portfolios\_select\_mp\_structure\_for\_accounts.sql to retrieve historical data in order to be presented in th UI

2|| UI Strategy changing history table => Created table component AppStrategyStructureHistTable to show historical data. Updated middleware layer and etc

3|| UI Strategy Form => Added strategy changing history table to the main strategy form. History is retrieved based on strategy id and updated onChange

4|| UI show snapshot of historical model portfolio on a given date =>

Using Main Strategy Form user is able to view historical model portfolio on the requested date and diffrence between weights on the requested date and current positons.

Function has been created without requesting data from DB. All data is beeing exchanged between components using services, streams and observables. Nessesary historical data is beeing trasferred from history table to the strategy form where it is porcessed and child structure table is updated.

Current data is stored inside component and a user is able to restore it witout requesting DB

--------------------------------------------------------------------------------------------

09/02/2024- 11/02/2024

1|| Transaction Types Dictionary UI&DB =>

Table component AppaIAccTransactionTypesTable has been created to retrieve/modify data regarding Transaction Types

Form component AppAccTransactionTypesFormComponent has been created to modify Transaction Type. Transaction Type must be unique by combination of type and code. User is able to block certain transaction type from manual editing. The option is introduced to protect integrity of data generated automatically and which is usually dependant on other axiuallary tables

2|| Accounting related interfaces have been transferred to the separate file. All dependant components and services have been updated accordingly

3|| Any type refactoring => Started getting rid of any type and replace it with correct types

--------------------------------------------------------------------------------------------

12/02/2024 - 17/02/2024

1|| Accounting schemes UI & DB =>

Created DB table bcSchemesParameters to store all parameters which could be used to create new accounting schemes. List of parameters is determined by function responsible for handling given buisness process. A parameter is similar to an api variable which is replaced by real value from business object when function is executed by the user

Introduced new object - "Process" and created respective DB table bcSchemesProcesses. Process object is used to make a link between given buisness process (function/module responsible for it) and list of parameters availble for the process

Create new table component AppAccSchemesParamsTable to show all availble parameters for the given process. The component is availble in the form of modify acccounting scheme. User can copy parameter to the clipboard by making a double click on it.

Table component AppAccSchemesAL\_Table has been introduced to retrieve/modify accounting account schemes data (entries stored bAccountTransaction and affect clients accounts)

Form component AppAccSchemesAL\_FormComponent has been created to modify accounting account schemes.

Table component AppAccSchemesLL\_Table has been introduced to retrieve/modify accounting ledger schemes data (entries stored bLedgerTransactions and don't affect clients accounts)

Form component AppAccSchemesLL\_FormComponent has been created to modify accounting ledger schemes.

Parameters are added to the scheme in the format of ${parameter\_name:raw}. Parameters could be used to customize entries attributes such as entry's details, entry amount, date and accounts corresponding.

There are two options to set the accounts corresponding in the accounting scheme. User can choose/set static account number from the existed accounts or insert a parameter linked to transaction's accounting objects. For example, client cash account is available through ${pAccontId} parameter while clients trades are being processed

Every process id is linked to the scheme id. All transactions have to be generated during the function exuction of the given process must have the same scheme id inside accounts related schemes and inside ledger schemes as well. The function retrieves all accounting schemes with the process's scheme id from bLedgerTransactions and bAccountTransaction

Implemented UI to retrive account number by id inputed by a user. If account hasn't been found then text input stored since text pattern could be used as parameter in accounting scheme and be populated by respective module while entries are created

New parameter counterparty code has been added to the api of SellTrade/BuyTrade acconting process

2|| Accounting schemes interfaces have been transferred to the separate file. All dependant components and services have been updated accordingly

3|| Any type refactoring => Refactored any type in allocation.service.ts

4|| Rewrite cached data when it has been changed => Introduced new function to the indexDB.service.ts to update cache data (replace cache with up to date snapshot). New function is used to update transactions types cached when user updates respective data

--------------------------------------------------------------------------------------------

19/02/2024 - 26/02/2024

1|| Refactored functionality to load currency rates from site of russian central bank:

The function has been moved from the component to the currency service

The function has been rewritten using RxJS library. All nested conditions have been replaced with RxJS operators inside the pipe

Timeout has been set for respective http get request and add hadler to process threpassing of the limit

In order to prevent a express server crash, when cbr site is unreacheable due to a bad or no connection, error hadler has been connected to the http get request. Error hadnler console log an error and send error to the client UI where is handled accordingly

Refactored UI to make it more intuitive

2|| Refactored any type to the secure ones in the following components:

allocation-table.component.ts,

orders-table.component.ts,

swift-items-table.component.ts,

swifts-incoming-table.component.ts,

accounting-trades.service.ts,

allocation.service.ts,

handling-entry-processing.service.ts,

tree-menu.service.ts,

trades-service.service.ts,

trade-table.component.ts,

fees-handling.service.ts,

currencies-data.service.ts,

currencies-data-table.component.ts

ToDo

|| Store histrory of a portfolio changing strategies

\* Fix: Allocation acconting - FIFO/Accounting => Roll back FIFO/Accounting when other fails

|| Revenue Factor Chart add info about management and perfomance fees deducted

|| FIFO Bug possible=> Case of simultaneously two trades which should be netted on FIFO are sent in FIFO function.

Idea is to sort processing batch according to the FIFO guidelines and execute FIFO function synchronously

|| Investment restrictions

|| any

|| Check UserAccess model for UI, espesically the latest components

|| DB access roles

|| Hurdle Performance fee and Cash In/Outs -??? How to correct start and end NVP?

\*Strategy tab component is not being reused and is rendered when reopened

\*Performance Chart shows graph from the last selected object if new object is reused and there's no need to reload it