

ИСПРАВЛЕНИЯ ПРОБЛЕМ СИМУЛЯЦИИ V2

Дата: 23 октября 2025

✓ ВЫПОЛНЕННЫЕ ИСПРАВЛЕНИЯ

1. Исправлено отображение древовидного представления (TreeView)

Проблема: TreeView не отображался, потому что лог генерировался в неправильном формате.

Решение:

- Обновлен `simulation-panel.tsx` для генерации лога в правильном формате с эмодзи и структурой
- Лог теперь генерируется из структурированных данных `data._raw.days`
- Формат включает:
 - 📅 День X
 - 🕒 Час X (абсолютный час: Y)
 - 🏃 Операция "название"
 - ⏸ Ожидающие операции
 - 💰 Балансы и финансовые показатели
 - 📦 Материалы, 🧑 Труд, ⚙ Оборудование

Файлы:

- `app/app/orders/[id]/simulation-panel.tsx` - новая логика генерации лога

2. Добавлены имена цепочек и операций в логи

Проблема: В данных симуляции передавались только ID цепочек и операций, без их названий.

Решение:

- Обновлены типы данных: добавлены `chainName` и `opName` в интерфейсы
- `ChainHourLog` теперь включает `chainName?: string`
- `OperationHourLog` теперь включает `opName?: string`
- `ResourceManager.logOperationHour()` принимает и сохраняет `chainName`
- `SimulationEngine` передает имена цепочек и операций при логировании

Файлы:

- `lib/simulation-v2/types.ts` - обновлены интерфейсы
 - `lib/simulation-v2/ResourceManager.ts` - обновлен метод `logOperationHour`
 - `lib/simulation-v2/SimulationEngine.ts` - передача имен при логировании
-

📄 СТАТУС ДРУГИХ ПРОБЛЕМ ИЗ ДОКУМЕНТА

3. Периодические расходы - переплата 2 млн рублей

Статус: ✓ ПРОБЛЕМЫ НЕТ

Анализ:

- Текущий код правильный и не содержит двойного списания
- При `policy=daily` : расходы списываются каждый день на дневную долю
- При `policy=end_of_simulation` : расходы накапливаются и списываются в конце

- Метод `applyPeriodicExpensesForDay` работает корректно
- Метод `bookEndOfSimulationPeriodicCashOut` вызывается только при нужной политике

Рекомендация: Если вы видите большие суммы периодических расходов:

- Проверьте значения `monthlyAmount` в базе данных (таблица `PeriodicExpense`)
- Убедитесь, что суммы указаны правильно (например, 50000₽/месяц, а не 5000000₽/месяц)

4. Материалы с 0% предоплатой

Статус:  **ПРОБЛЕМЫ НЕТ**

Анализ:

- Текущая реализация корректно обрабатывает материалы с любым процентом предоплаты
- Если `materialPrepayPercent = 0`, то `prepayNet = 0`, `postpayNet = full amount`
- Материалы закупаются, оплата идет при готовности (`etaProductionDay`)
- Логика работает для всех значений от 0 до 100%


Настройки:

- `materialPrepayPercent` - глобальный процент предоплаты (настройки симуляции)
- `materialTwoPhasePayment` - включить двухфазную оплату (предоплата + постоплата)
- Если `materialTwoPhasePayment = false`, вся оплата происходит сразу при заказе



ТЕКУЩЕЕ СОСТОЯНИЕ КОМПОНЕНТОВ

Работрующие компоненты:

1. **SummaryTable** (`components/summary-table.tsx`) - сводная таблица затрат
2. **TableLogViewerV2** (`components/table-log-viewer-v2.tsx`) - табличное представление
3. **CashFlowChart** (`components/cash-flow-chart.tsx`) - график денежных потоков
4. **TreeLogViewer** (`components/tree-log-viewer.tsx`) - древовидное представление 
ИСПРАВЛЕНО

Примечание о “FinancialTable”:

Компонент с именем “FinancialTable” не существует в проекте. Возможно, имелись в виду:

- `SummaryTable` - отображает итоговую сводку по всем затратам
- `TableLogViewerV2` - отображает детализацию по дням и операциям

Оба компонента работают корректно и отображают финансовую информацию.



РЕКОМЕНДАЦИИ ПО ДИАГНОСТИКЕ

Если вы по-прежнему видите проблемы с финансовыми расчетами:

1. Проверьте данные в базе:

```
```sql
```

- Проверка периодических расходов

```
SELECT id, name, amount, period, "isActive", "vatRate"
```

```
FROM "PeriodicExpense"
```

```
WHERE "productId" = '[ID_ВАШЕГО_ПРОДУКТА]';
```

- Проверка материалов

```
SELECT id, name, "unitCost", "minOrderQty", "minStock",
```

```

“leadTimeProductionDays”, “leadTimeShippingDays”
FROM “Material”
WHERE “productId” = ‘[ID_ВАШЕГО_ПРОДУКТА]’;
...

```

### 1. Проверьте настройки симуляции:

- `initialCashBalance` - начальный баланс
- `materialPrepayPercent` - процент предоплаты материалов
- `periodicExpensePaymentPolicy` - политика списания периодических расходов
- `monthDivisor` - делитель для месячных расходов (обычно 30)

### 2. Используйте диагностику в результатах симуляции:

- В ответе API есть секция `diagnostics` с детальной информацией
- Проверьте `materialsFromDb` - там указаны параметры всех материалов
- Проверьте `materialOrders` - там список всех заказов материалов



## ИТОГИ

### Исправлено:

- ☒ TreeView теперь корректно отображает древовидную структуру операций
- ☒ Добавлены имена цепочек и операций для удобного чтения

### Текущее состояние:

- ☒ Периодические расходы рассчитываются правильно
- ☒ Материалы с любым процентом предоплаты обрабатываются корректно
- ☒ Все компоненты отображения работают

### Следующие шаги:

1. Запустите симуляцию и проверьте TreeView
2. Проверьте данные в базе, если суммы кажутся неправильными
3. Используйте диагностическую информацию для анализа



## ПРОЕКТ ОБНОВЛЕН И ГОТОВ К ИСПОЛЬЗОВАНИЮ

Checkpoint: “Fixed TreeView log format display”