ERP/1: Облік

Техноробочий проект

на розробку та впровадження
підсистеми обліку, кадрів
та заробітної плати

Згідно з Постановою КМУ № 205 від 21.02.2025
Формалізація вимог за стандартом ISO/IEC/IEEE 29148:2018

Зміст

Зміст

1. Загальні відомості про засіб інформатизації

1.1. Найменування та підстави для розробки

Найменування засобу інформатизації: Підсистема «ERP/1: Облік» (далі — Модуль «Облік»).
Відомості про замовника: ТОВ "Системи Електронного Урядування"в ТОВ "Криптографічні телесистеми".
Підстави для виконання робіт:

- Бюджетний кодекс України;
- Закон України «Про бухгалтерський облік та фінансову звітність в Україні»;
- Постанова Кабінету Міністрів України від 21 лютого 2025 року № 205 «Деякі питання створення, адміністрування та забезпечення функціонування засобу інформатизації».

1.2. Призначення та цілі створення засобу

Модуль «Облік» призначений для комплексної автоматизації фінансової та господарської діяльності з можливістю багатовимірного аналітичного обліку, розрахунку заробітної плати та складання регламентованих звітів.
Цілі створення засобу:

- Автоматизація ведення Головної книги та бухгалтерських проводок;
- Складання оборотного балансу та фінансових звітів у реальному часі;
- Розрахунок заробітної плати та податків з особових рахунків працівників;
- Управління матеріально-технічним забезпеченням, закупівлями та складом ТМЦ;
- Контроль виконання кошторисів за кодами КЕКВ;
- Забезпечення автентифікації та підписання відомостей за допомогою КЕП (CAdES-X Long).

1.3. План-графік виконання етапів робіт

1. Етап 1: Проектування та погодження архітектури — розробка та погодження UML/Figma специфікацій. (1 місяць).
2. Етап 2: Створення схем даних — реалізація Erlang records, створення таблиць Mnesia та RocksDB сховищ. (1.5 місяці).
3. Етап 3: Розробка бізнес-процесів — реалізація BPE DSL для ЗП, проводок та закупівель. (2 місяці).
4. Етап 4: Інтеграційні сервіси — налаштування обміну з Є-Казна, ДПС, ПФУ. (1.5 місяці).
5. Етап 5: Верифікація та тестування — Unit, Integration тестування, UAT приймання. (1 місяць).

2. Відомості про робочий процес та умови експлуатації

2.1. Опис бізнес-процесів (BPMN / FSM)

2.1.1. Процес проводки первинного документа

1. StartPosting: Створення проекту документа бухгалтером.
2. EnterData: Введення реквізитів (сума, рахунки дебету/кредиту, код КЕКВ).
3. ValidateDoubleEntry: Автоматична перевірка валідатором подвійного запису. У разі успіху — перехід до `CheckBudget`, інакше — повернення до `EnterData`.
4. CheckBudget: Автоматичний контроль відповідності лімітам видатків за КЕКВ.
5. SignDoc: Накладання КЕП бухгалтера на первинний документ.
6. PostToLedger: Автоматичне проведення суми по рахунках Головної книги.
7. PostingCompleted: Фіксація проводки.

2.1.2. Процес розрахунку заробітної плати

1. StartPayroll: Запуск розрахункового періоду за місяць.
2. CollectTimesheets: Завантаження та затвердження табелів робочого часу працівників.
3. CalculateTaxes: Автоматичний розрахунок нарахувань, утримань ПДФО, військового збору та ЄСВ.
4. ApproveRegister: Погодження відомості розрахунку Головним бухгалтером.
5. QESSign: Накладання КЕП керівника закладу на відомість виплати.
6. DisburseFunds: Формування платіжного реєстру для відправки в Є-Казна / банк.
7. PayrollCompleted: Закриття періоду.

2.1.3. Процес планування закупівель

1. StartProcurement: Запуск процедури планування закупівлі ТМЦ.
2. DraftRequest: Створення заявки на закупівлю із зазначенням кодів ТМЦ та КЕКВ.
3. VerifyFunds: Автоматична перевірка наявності кошторисних асигнувань.
4. ApproveByChief: Погодження закупівлі керівництвом.
5. GenerateContract: Автоматичне формування договору в реєстрі укладених угод.
6. ProcurementCompleted: Завершення етапу планування.

2.1.4. Процес підготовки та подання звітності

1. StartReporting: Запуск формування звітного періоду (місяць, квартал, рік).
2. RunConstructor: Збирання даних з Головної книги та кадрового обліку.
3. ValidateData: Контроль взаємозв'язку показників форм звітності.
4. QESSignReport: Накладання КЕП посадових осіб.

5. SubmitToPortal: Відправка звіту через інтеграційне API до казначейського або податкового порталу.
6. ReportAccepted: Отримання квитанції про успішне прийняття звіту.

2.2. Опис ролей та прав доступу (ABAC)

- Правило 1 (Бухгалтер):

$$\text{subject.roles contains 'Accountant'} \land \text{object.entity\_type} == \text{'PrimaryDocument'}$$

$$\land\ \text{object.department\_id} == \text{subject.department\_id} \land \text{object.status} == \text{'draft'}$$

$$\rightarrow \text{Permit: read, write, sign.}$$

- Правило 2 (Кадровик):

$$\text{subject.roles contains 'HR'} \land (\text{object.entity\_type} == \text{'EmployeeRecord'} \lor \text{object.entity\_type} == \text{'TimesheetEntr}$$

$$\rightarrow \text{Permit: read, write, sign. Deny: delete.}$$

- Правило 3 (Комірник):

$$\text{subject.roles contains 'Storekeeper'} \land \text{object.entity\_type} == \text{'WarehouseItem'}$$

$$\rightarrow \text{Permit: read, write (тільки прибуткові/видаткові ордери).}$$

- Правило 4 (Контроль подвійного запису):

$$\text{object.entity\_type} == \text{'JournalPosting'} \land \text{debit\_sum} == \text{credit\_sum}$$

$$\rightarrow \text{Permit: post.}$$

- Правило 5 (Кошторисний контроль):

$$\text{object.entity\_type} == \text{'PrimaryDocument'} \land \text{object.amount} \leq \text{budget.available\_limit}$$

$$\rightarrow \text{Permit: approve.}$$

2.3. Умови експлуатації

- Середовище виконання: Erlang/OTP версії 26 та вище.
- База даних: Erlang Mnesia як єдине native транзакційне сховище. Використання реляційних SQL баз даних не допускається.
- Сховище медіа-об'єктів (первинних скан-копій): S3-сумісне сховище або RocksDB через бібліотеку KVS.

3. Інформаційне та програмне забезпечення

3.1. Інформаційне забезпечення (Реквізити)

Усі об'єкти описуються Erlang-рекордами:

```
-record(primary_document, {
    id, date, document_number, type, counterparty,
    debit_account, credit_account, amount, vat, status = draft
}).

-record(journal_posting, {
    id, date, document_number, description, debit_account,
    credit_account, amount, kekv, department, status = draft
}).

-record(ledger_account, {
    id, code, name, opening_balance, debit_turnover,
    credit_turnover, closing_balance
}).

-record(employee_record, {
    id, full_name, personnel_number, department, position,
    hire_date, salary, status = active
}).

-record(timesheet_entry, {
    id, employee_name, department, days_worked, days_absent,
    overtime_hours, total_hours, status = draft
}).

-record(payroll_line, {
    id, employee_name, department, position, gross_pay,
    pit, military_tax, pension_fund, other_deductions,
    net_pay, status = calculated
}).

-record(supply_contract, {
    id, contract_number, date, supplier, total_value,
    executed_amount, stages_completed, total_stages,
    end_date, status = draft, kekv
}).

-record(warehouse_item, {
    id, code, name, unit, quantity, price, total_value,
    min_stock, warehouse, stock_level = normal
}).
```

Повнотекстові специфікації реквізитів: наведено в додатку А до Технічних вимог (див. [acc.tex](file:///Users/tonpa/depot/erpuno/erp.uno/acc/acc.tex)).

3.2. Програмне забезпечення (Реалізація)

3.2.1. bpe_bookkeeping_posting.erl (Процес проведення документа)

```
-module(bpe_bookkeeping_posting).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Primary Document Posting',
        beginEvent = 'StartPosting',
        endEvent = 'PostingCompleted',
        tasks = [
            #beginEvent{name='StartPosting'},
            #userTask{name='EnterData', module=?MODULE},
            #serviceTask{name='ValidateDoubleEntry', module=?MODULE},
            #serviceTask{name='CheckBudget', module=?MODULE},
            #userTask{name='SignDoc', module=?MODULE},
            #serviceTask{name='PostToLedger', module=?MODULE},
            #endEvent{name='PostingCompleted'}
        ],
        flows = [
            #sequenceFlow{name='1', source='StartPosting', target='EnterData'},
            #sequenceFlow{name='2', source='EnterData',
                          target='ValidateDoubleEntry'},
            #sequenceFlow{name='3', source='ValidateDoubleEntry',
                          target='CheckBudget', condition="validation_ok"},
            #sequenceFlow{name='4', source='ValidateDoubleEntry',
                          target='EnterData', condition="validation_failed"},
            #sequenceFlow{name='5', source='CheckBudget',
                          target='SignDoc', condition="budget_ok"},
            #sequenceFlow{name='6', source='CheckBudget',
                          target='EnterData', condition="budget_overlimit"},
            #sequenceFlow{name='7', source='SignDoc', target='PostToLedger'},
            #sequenceFlow{name='8', source='PostToLedger',
                          target='PostingCompleted'}
        ]
    }.

action({serviceTask, 'ValidateDoubleEntry'}, Proc) ->
    case acc_validator:validate_posting(Proc) of
        ok -> {reply, Proc, "validation_ok"};
        _  -> {reply, Proc, "validation_failed"}
    end;
action(_, Proc) -> {reply, Proc}.
```

3.2.2. bpe_payroll_calculation.erl (Процес розрахунку ЗП)

```
-module(bpe_payroll_calculation).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Monthly Payroll Calculation',
```

```
        beginEvent = 'StartPayroll',
        endEvent = 'PayrollCompleted',
        tasks = [
            #beginEvent{name='StartPayroll'},
            #serviceTask{name='CollectTimesheets', module=?MODULE},
            #serviceTask{name='CalculateTaxes', module=?MODULE},
            #userTask{name='ApproveRegister', module=?MODULE},
            #userTask{name='QESSign', module=?MODULE},
            #serviceTask{name='DisburseFunds', module=?MODULE},
            #endEvent{name='PayrollCompleted'}
        ],
        flows = [
            #sequenceFlow{name='1', source='StartPayroll',
                          target='CollectTimesheets'},
            #sequenceFlow{name='2', source='CollectTimesheets',
                          target='CalculateTaxes'},
            #sequenceFlow{name='3', source='CalculateTaxes',
                          target='ApproveRegister'},
            #sequenceFlow{name='4', source='ApproveRegister',
                          target='QESSign'},
            #sequenceFlow{name='5', source='QESSign',
                          target='DisburseFunds'},
            #sequenceFlow{name='6', source='DisburseFunds',
                          target='PayrollCompleted'}
        ]
    }.
action(_, Proc) -> {reply, Proc}.
```

3.2.3. bpe_procurement_planning.erl (Процес планування закупівель)

```
-module(bpe_procurement_planning).
-include("bpe.hrl").
-compile(export_all).

def() ->
    #process{
        name = 'Procurement Planning',
        beginEvent = 'StartProcurement',
        endEvent = 'ProcurementCompleted',
        tasks = [
            #beginEvent{name='StartProcurement'},
            #userTask{name='DraftRequest', module=?MODULE},
            #serviceTask{name='VerifyFunds', module=?MODULE},
            #userTask{name='ApproveByChief', module=?MODULE},
            #serviceTask{name='GenerateContract', module=?MODULE},
            #endEvent{name='ProcurementCompleted'}
        ],
        flows = [
            #sequenceFlow{name='1', source='StartProcurement',
                          target='DraftRequest'},
            #sequenceFlow{name='2', source='DraftRequest',
                          target='VerifyFunds'},
            #sequenceFlow{name='3', source='VerifyFunds',
                          target='ApproveByChief', condition="funds_available"},
```

```
            #sequenceFlow{name='4', source='VerifyFunds',
                          target='DraftRequest', condition="insufficient_funds"},
            #sequenceFlow{name='5', source='ApproveByChief',
                          target='GenerateContract'},
            #sequenceFlow{name='6', source='GenerateContract',
                          target='ProcurementCompleted'}
        ]
    }.
action(_, Proc) -> {reply, Proc}.
```

3.2.4. acc_validator.erl (Модуль бізнес-валідацій)

```
-module(acc_validator).
-export([validate_posting/1, check_budget_limit/3]).

-record(posting, {debit_amount = 0.0, credit_amount = 0.0}).

validate_posting(Postings) ->
    DebitSum = lists:foldl(fun(P, Acc) -> Acc + P#posting.debit_amount end, 0.0, Postings),
    CreditSum = lists:foldl(fun(P, Acc) -> Acc + P#posting.credit_amount end, 0.0, Postings),
    case DebitSum == CreditSum of
        true -> ok;
        false -> {error, balance_mismatch}
    end.

check_budget_limit(KEKV, Amount, AvailableLimit) ->
    case Amount =< AvailableLimit of
        true -> ok;
        false -> {error, {budget_overlimit, KEKV}}
    end.
```

4. Вимоги до засобу за ISO/IEC/IEEE 29148

4.1. Функціональні вимоги

- [REQ-ACC-FUN-001] Модуль «Облік» повинен забезпечувати ведення плану рахунків, класифікації КЕКВ та довідників ТМЦ.
- [REQ-ACC-FUN-002] Модуль «Облік» повинен виконувати автоматичну перевірку балансу дебету та кредиту для кожної господарської проводки.
- [REQ-ACC-FUN-003] Модуль «Облік» повинен підтримувати кошторисний контроль — блокувати проведення первинних документів, якщо сума видатків за кодом КЕКВ перевищує ліміт асигнувань.
- [REQ-ACC-FUN-004] Модуль «Облік» повинен забезпечувати автоматичний розрахунок заробітної плати працівників за місяць на основі затверджених табелів обліку робочого часу.
- [REQ-ACC-FUN-005] Модуль «Облік» повинен автоматично розраховувати суми утримань ПДФО (18%), військового збору (1.5%) та нарахувань ЄСВ (22%) згідно з чинним законодавством України.
- [REQ-ACC-FUN-006] Модуль «Облік» повинен підтримувати підписання розрахункових та платіжних відомостей за допомогою КЕП CAdES-X Long.
- [REQ-ACC-FUN-007] Модуль «Облік» повинен забезпечувати автоматичний розрахунок оборотного балансу та відображення залишків по Головній книзі в режимі реального часу.
- [REQ-ACC-FUN-008] Модуль «Облік» повинен забезпечувати ведення особових справ співробітників, враховуючи кадрові накази та штатні оклади.
- [REQ-ACC-FUN-009] Модуль «Облік» повинен генерувати первинні документи (Акт, Накладна, Розрахунковий листок) у форматах PDF та XLSX згідно з регламентованими формами.
- [REQ-ACC-FUN-010] Модуль «Облік» повинен експортувати фінансові звіти до системи «Є-Звітність» за допомогою інтеграційного HTTPS REST API.

4.2. Вимоги до інтерфейсів та дизайну

- [REQ-ACC-UI-001] Користувацький інтерфейс повинен бути сумісним зі стандартом веб-доступності WCAG 2.1 AA.
- [REQ-ACC-UI-002] Інтерфейс користувача повинен функціонувати як Single Page Application (SPA), забезпечуючи оновлення фінансових моніторів через Secure WebSockets (WSS).
- [REQ-ACC-UI-003] Обмін даними між веб-клієнтом та сервером повинен використовувати архітектурний шаблон Data-on-Wire з передачею об'єктів у форматі JSON.

4.3. Вимоги до безпеки та захисту інформації

- [REQ-ACC-SEC-001] Автентифікація всіх користувачів підсистеми повинна здійснюватися виключно за допомогою кваліфікованого електронного підпису (КЕП) X.509.
- [REQ-ACC-SEC-002] Будь-які запити на читання та запис фінансових даних повинні перевірятися підсистемою контролю доступу на основі ABAC-атрибутів суб'єкта та об'єкта.
- [REQ-ACC-SEC-003] Система повинна передавати дані виключно через протокол шифрування TLS 1.3 з використанням безпечних криптографічних алгоритмів.
- [REQ-ACC-SEC-004] Оскільки реляційні SQL бази даних не використовуються, ризик SQL-ін'єкцій повністю відсутній.
- [REQ-ACC-SEC-005] Персональні дані працівників (особові справи, розрахунки ЗП) повинні бути захищені від несанкціонованого доступу згідно із Законом України «Про захист персональних даних».

4.4. Вимоги до продуктивності

- [REQ-ACC-PERF-001] Час розрахунку та відображення оборотного балансу по 50,000 операцій не повинен перевищувати 500 мс.
- [REQ-ACC-PERF-002] Пропускна здатність підсистеми Mnesia повинна бути не менше 500 фінансових проводок на секунду на один серверний вузол.
- [REQ-ACC-PERF-003] Час відкриття та відображення реєстру документів на 1000 записів на клієнті не повинен перевищувати 1 секунди.

4.5. Вимоги до логування, тестування та супроводу

- [REQ-ACC-LOG-001] Усі операції зміни фінансових станів, коригування проводок та входу користувачів повинні реєструватися у захищеному журналі аудиту (Audit Log).
- [REQ-ACC-LOG-002] Логи аудиту повинні передаватися у форматі JSON до Elasticsearch (стек ELK) в режимі реального часу.
- [REQ-ACC-ADM-001] Покриття вихідного коду автоматичними Unit-тестами та інтеграційними тестами надійності повинно бути не менше ніж 80%.
- [REQ-ACC-ADM-002] Поставка програмного забезпечення повинна містити повний пакет супровідної документації (Настанова користувача, Настанова адміністратора) згідно з ДСТУ 3008:2015.