ЗМІСТ

ВСТУП

Формування стабільної банківської системи України, здатної забезпечити кредитування реального сектора економіки, є однією з головних завдань українського суспільства. Стійкість банківської системи в умовах ринкової економіки багато в чому залежить від результатів наукового аналізу пройденого етапу її становлення, проблем формування та адаптації до нових умов діяльності.

Ця дипломна робота націлена на рішення найпоширеніших проблем серед сайтів інших банків, а саме своєчасне оновлення, а також легкість оновлення інформації на сайті. Легкий, лаконічний, красивий і в той же час зручний і інтуїтивно зрозумілий дизайн сайту. Висока швидкість оновлення і роботи веб-додатки не залежно від того де знаходиться користувач і який пристрій він використовує для відвідування сайту банку.

Сучасною проблемою всіх банківських систем є їх висока вразливість до злому з боку шахраїв. Алгоритми повинні оновлюватися якомога частіше і краще перевірятися під час їх тестування. У цьому додатку буде приділятися більше часу до безпеки використання веб-додатки.

Головна причина розробки веб-додатку банку є зручність використання функціоналу банку в будь-якій точці світу з будь-якого пристрою, що має доступ до мережі інтернету. Якщо користувач зможе користуватися банком де захоче, це безсумнівно приверне нових користувачів банку.

Зараз доступ до Інтернету має кожна людина, з цього створення веб-додатка банку до яких може користуватися кожен є головним аргументом актуальності даної роботи.

WEB-додаток банківської системи – це програма створена для полегшення взаємодії між користувачем цього банку і самим банком. Ця програма дозволить користувачам взаємодіяти зі своїми рахунками або картами в цьому банку, а також дозволить завести обліковий запис в банку,

завести карту банку, здійснювати транзакції і перекази між картками користувачів банку.

Унікальність даного веб-додатку полягає в тому, що тільки цей додаток здатний взаємодіяти з банком, до якого він підключений. Веб додаток – це зручний ресурс, який дозволить користувачам взаємодіяти з ним в будь-якій точці світу з абсолютно будь-якого пристрою, який підтримує вихід в інтернет.

Додаток дозволить користувачам з легкістю завести картку в банку, поповнити карту, перевести кошти з однієї картки на іншу, взяти кредит в цьому банку, створити карту для виплат або ощадну картку, відстежувати історію транзакцій і платежів, заблокувати карту в разі втрати або крадіжки, зв'язатися з представником банку для усунення проблем або отримання відповідей на питання.

У додатку є новинна стрічка, за допомогою якої користувачі банку будуть дізнаватися про нові акції, нововведення, вигідних вкладень, попередження про шахрайство або хакерських атаках.

Інтуїтивно зрозумілий інтерфейс дозволить користувачам швидко освоїться в додатку і з легкістю розуміти що вони роблять.

Додаток розробляється на мові PHP, тому що ця мова є одним з найпопулярніших мов програмування веб додатки, і дозволяє реалізувати всі функції, які необхідні для застосування банківської системи.

1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ

1.1 Формулювання мети і постановка задачі

Веб-додаток розробляється для автоматизації роботи того банку в який буде впроваджений цей сайт. Веб-додаток повинен містити ту інформацію і ті можливості які існують і доступні в цьому банку. Веб-додаток повинен бути простий у використанні і легко запам'ятовується користувачам. Веб-додаток повинен мати можливість якомога швидко і часто оновлюватися з програмної точки зору, алгоритми переказів повинні постійно удосконалюватися і змінюватися що б бути максимально безпечним і недоступним для шахраїв.

1.2 Аналіз предметної області

Для чого потрібний веб-додаток банку:

За допомогою цього веб-додатку у користувача завжди є можливість перевірити свої рахунки онлайн, створювати нові рахунки або закривати старі, здійснювати перекази між своїми рахунками або на рахунок іншої людини для оплати чого-небудь. Так само у користувача є можливість переглядати історію скоєних транзакцій.

Для кого необхідний це веб-додаток:

1. для людей, які люблять порядок в грошах;
2. для бізнесменів, які часто роблять переклади;
3. для тих, хто хоче бути впевненим що їхні гроші в безпеці.
4. для людей, які хочуть, щоб їхня робота з грошима була максимально автоматизована.

Простіше кажучи, інтернет-банк гарантує постійний доступ до своїх коштів де б ви не були, а також поповнення ваших коштів і здійснення платежів за допомогою них. Ви можете носити з собою величезну суму грошей, і вони не будуть займати багато місця. При цьому ви можете не

турбуватися про збереження ваших коштів, так як ви не зможете їх втратити на відміну від готівкових коштів і у вас їх не зможуть вкрасти, адже система і додаток постійно оновлюються, а їх безпеку зростає з кожним днем.

У веб-додатку банку повинні бути такі функції:

1. створення облікового запису;
2. створення рахунку в банку;
3. відстеження коштів на рахунку;
4. перекази між рахунками;
5. перегляд історії транзакцій;
6. можливість закрити рахунок;
7. поповнення рахунку;
8. зняття «готівки» (в додатку «термінал»);
9. зміна пароля або поштової скриньки;

1.3. Огляд існуючих аналогів

На сьогоднішній день в Україні існує велика кількість банківських систем і сайтів, які їх підтримують. Лідером серед всіх банківських систем є «Приват-банк». Це дуже надійна і швидка система яка постійно оновлюється і покращує безпеку здійснення маніпуляцій з картами і рахунками. Ця система розробляється вже досить тривалий час, за яке встигла зазнати кардинальні зміни і в плані програмної точки зору, і в плані поліпшення безпеки. Головними відмінностями цієї системи є миттєві переклади без помилок з програмної точки, захист переводів за допомогою шифрування, інтуїтивно зрозумілий інтерфейс, багатомовність. Система банку «Приват банк» є саме тією системою, на яку варто рівнятися.

Так само є ще не мало добре побудованих банківських систем. І у кожної системи є свої переваги і унікальні можливості. Прикладами таких банківських систем можуть бути такі банки як: «Альфа банк», «А банк», «Мега-банк», «ПУМБ», «Укрсиббанк». У кожній системі цих банків присутній

унікальний дизайн і інтерфейс користувача. Так чи інакше всі ці банківські системи по-різному захищені від зломів і шахрайських операцій. Всі ці банки часто оновлюються як в програмному плані, так і в плані дизайну. Завдяки актуальності інформації на веб ресурсах цих банків знайдеться велика кількість різних людей, які їм довіряють і рекомендують їх, в тому числі і іноземних користувачів.

Використовувані в документі терміни, визначення та скорочення наведені в таблиці 1.1.

Таблиця 1.1 – Визначення, позначення і скорочення

Терміни / Скорочення	Визначення / Повне найменування
БД	База даних (БД) - це організована структура, призначена для зберігання, зміни і обробки взаємозалежної інформації, переважно великих обсягів.
ПЗ	Програмне забезпечення-це сукупність програм, використовуваних на комп'ютері і забезпечують функціонування його апаратних засобів, виконання різних за-дач користувача, а також розробку і налагодження нових програм.
ОС	Операційна система- це перший і основний набір програм, що завантажується в комп'ютер.
PHP	PHP - мова програмування веб-додатків.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1 Вимоги до програмного забезпечення

Функціональні вимоги

Вимоги для користувача:

- реєстрація користувача;
- вхід користувача в особистий кабінет;
- перегляд профілю;
- редагування особистого профілю;
- створення карт;
- перегляд карт;
- переказ коштів на свою карту;
- переказ коштів на іншу карту за номером;
- видалення карти;
- поповнення карти через термінал;
- виведення готівки через термінал;
- зміна паролю та поштової скриньки;
- відновлення паролю;

Вимоги для адміністратору сайту:

- вхід адміністратора в особистий кабінет для управління веб Додатком;
- просмотр статистики сайту(кількість користувачів);
- додавання новин на головну сторінку;
- відповідання на технічні питання користувачів.

Нефункціональні вимоги

- зручний інтерфейс користувача;
- простота використання;
- надійність;

– можливість розширення;

– функціональна повнота;

– переносимість веб-додатку на інші сервери;

– всі сторінки веб-додатку повинні відкриватися не довше 2 секунд

Мови сайту

Головна мова сайту – російська.

2.2 Побудова схеми БД

Побудова схем БД

ER-діаграма (ER-модель) – це модель даних, що дозволяє описувати концептуальні схеми предметної області. ER-модель використовується при високорівневої проектуванні баз даних. З її допомогою можна виділити ключові сутності і позначити зв'язки, які можуть встановлюватися між цими сутностями. На рисунку 2.1 представлено Таблиці БД.

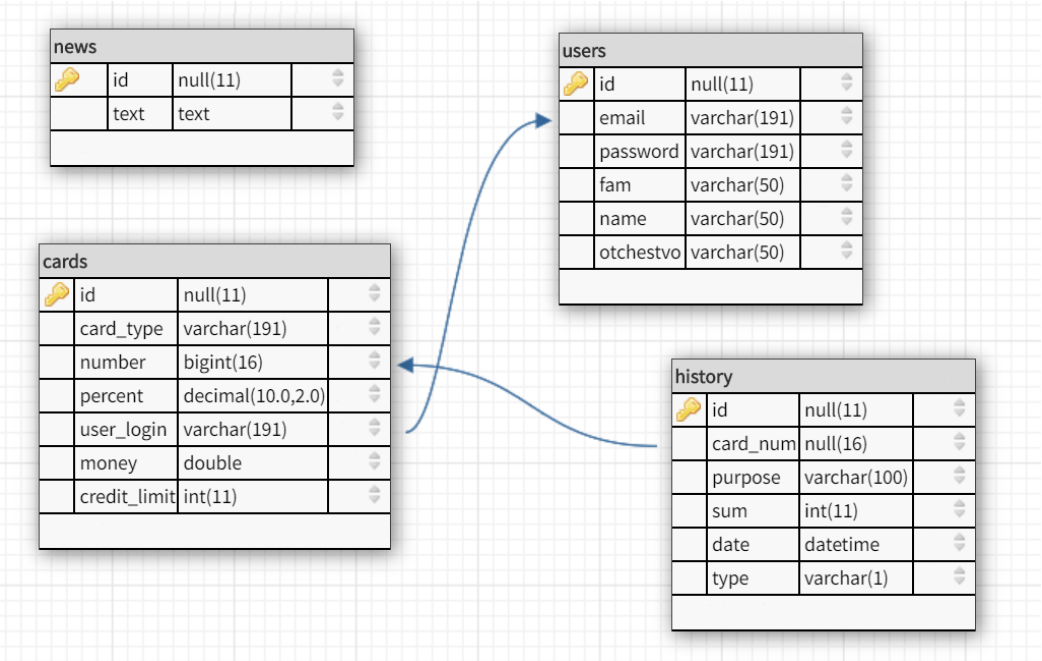

Рисунок 2.1 – Таблиці БД

2.3 Побудова UML діаграм

Діаграма використання (use-case) дозволяє наочно побачити, як користувачі будуть взаємодіяти з веб-додатком банківської системи (рис.2.2).

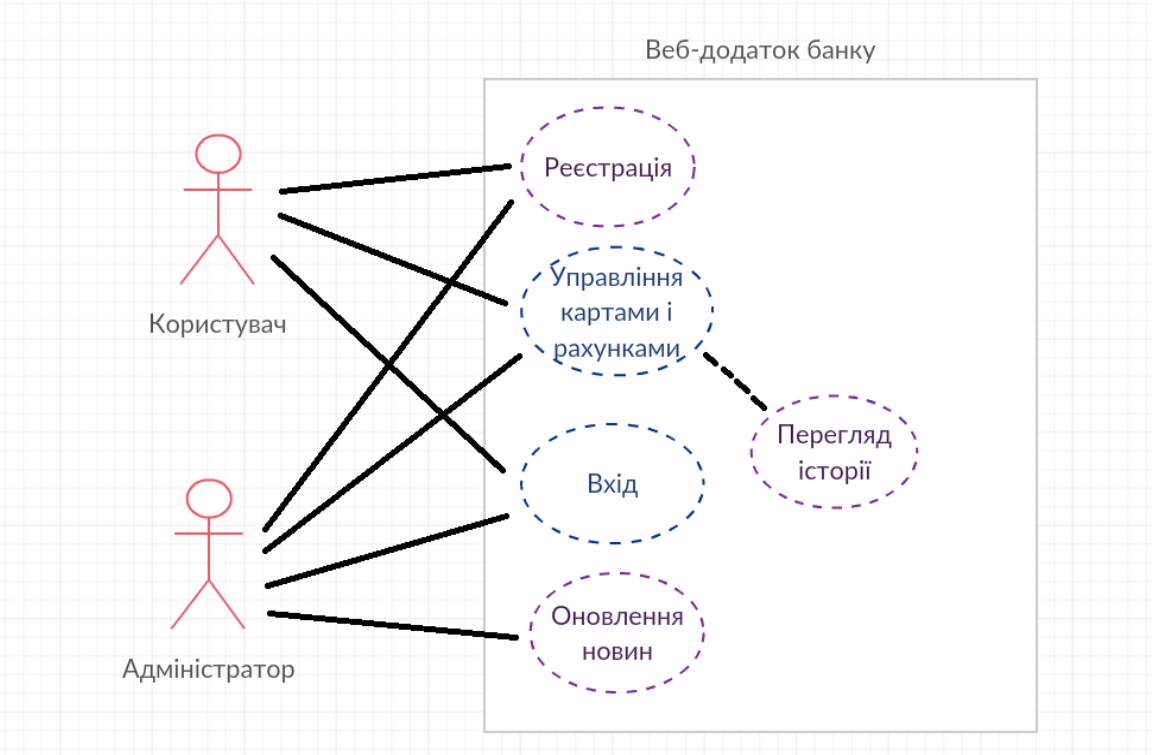

Рисунок 2.2 – Діаграма використання

3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Вибір мови та середовища розробки

При створенні веб-додатку банківської системи використовувалися мови розмітки HTML та CSS, а також мова веб-програмування PHP.

PHP – це скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-серверу. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів.

PHP інтерпретується веб-сервером в HTML-код, який передається на сторону клієнта. На відміну від таких скриптових мов програмування, як JavaScript, користувач не має доступу до PHP-коду, що є перевагою з точки зору безпеки але значно погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів які виконаються вже на стороні клієнта. PHP – мова, яка може бути вбудованою безпосередньо в html-код сторінок, які, в свою чергу коректно будуть оброблені PHP-інтерпретатором. Механізм PHP просто починає виконувати код після першої екрануючої послідовності (<?) і продовжує виконання до того моменту, коли він зустріне парну екрануючу послідовність (?>). Велика різноманітність функцій PHP дають можливість уникнути написання багаторядкових призначених для користувача функцій на C або Pascal. Наявність інтерфейсів до багатьох баз даних:

– в PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase.

– через стандарт відкритого інтерфейсу зв’язку з базами даних (Open Database Connectivity Standard – ODBC) можна підключатися до всіх баз даних, до яких існує драйвер. [4]

3.2 Вибір СУБД і опис її фізичної моделі

MySQL – це система управління реляційними базами даних. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається перевага у швидкості й гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість поєднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мова структурованих запитів, що використовується для доступу до баз даних. MySQL – це ПЗ з відкритим кодом. Застосовувати його і модифікувати може будь-хто. Таке ПЗ можна отримувати за допомогою Internet і використовувати безкоштовно. При цьому кожен користувач може вивчити вихідний код і змінити його у відповідності зі своїми потребами. MySQL складається з двох частин: серверної і клієнтської. Сервер MySQL постійно працює на комп'ютері. Клієнтські програми (наприклад, скрипти PHP) посилають серверу MySQL SQL-запити через механізм сокетів (тобто за допомогою мережевих засобів), сервер їх обробляє і запам'ятовує результат. Тобто скрипт (клієнт) вказує, яку інформацію він хоче отримати від сервера баз даних. Потім сервер баз даних посилає відповідь(результат) клієнтові(скрипту).

Структура MySQL трирівнева: бази даних-таблиці-записи. Бази даних і таблиці MySQL фізично представляються файлами з розширеннями frm, MYD, MYI. Логічно таблиця являє собою сукупність записів. А запису – це сукупність полів різного типу. Ім'я бази даних MySQL унікально в межах системи, а таблиці – в межах бази даних, поля – в межах таблиці. Один сервер MySQL може підтримувати одразу декілька баз даних, доступ до яких може розмежовуватись логіном і паролем. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди серверакомпілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому. [5]

3.3 Реалізація основних класів і методів

Програмна реалізація бази даних наведена у додатку А.

Даний проект містить такий набір папок:

1. tmp – папка в якій знаходиться бібліотека ReadBean PHP.
2. Css – папка для css-файлів.
3. Terminal – папка для программи терміналу.

Даний проект містить такі файли:

1. index.php – файл, в якому реалізовано відображення особистого профілю користувача.
2. options.php – файл з налаштуваннями.
3. db.php – підключення до бази даних.
4. Login.php – вихід користувача з веб-додатки.
5. Logout.php – вихід користувача з веб-додатка.
6. Signup.php – реєстрація користувача.

4. ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Планування тестів

Планування тестування представлено у вигляді тест-кейсів (таблиці 4.1-4.7).

Таблиця 4.1 – Тест-кейс№1

Назва:	Тест-кейс№1	
Функція:	Реєстрація	
Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Відкрити головну сторінку в додатку; 2. Натиснути на кнопку створення облікового запису;	Відкрилася сторінка реєстрації користувача	• пройдений
Кроки тесту:		
3. Заповнюємо всі поля на сторінці реєстрації;	Користувач успішно зареєстрований і його перекидає на сторінку входу.	• пройдений

Таблиця 4.2 – Тест-кейс№2

Назва:	Тест-кейс№2
Функція:	Створення карти користувача

Продовження таблиці 4.2

Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Відкрити головну сторінку в додатку; 2. Вибрати карту і ввести пін-код;	З'явилося підтвердження створення карти	• пройдений
Кроки тесту:		
3. Натиснути на кнопку створення карти;	Карта успішно створена і з'явилася в списку карт	• пройдений

Таблиця 4.3 – Тест-кейс№3

Назва:	Тест-кейс№3	
Функція:	Переказ коштів з картки на картку	
Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Відкрити головну сторінку в додатку; 2. У блоці переказу коштів між своїми картами заповнити всі поля; 3. Натиснути на кнопку перевести;	З'явиться інформація про комісію і кнопка підтвердження	• пройдений

Продовження таблиці 4.3

Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Кроки тесту:		
4. Натиснути на кнопку підтвердити;	Списано гроші з однієї картки та поповнено іншу картку на суму списання. Комісія пішла до фонду банку. У списку історій транзакцій з'явився запис про переказ коштів.	• пройдений

Таблиця 4.4 – Тест-кейс№4

Назва:	Тест-кейс№4	
Функція:	Перегляд історії транзакцій	
Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Відкрити головну сторінку в додатку;	Відкрилася сторінка з інструментами	• пройдений
Кроки тесту:		
2. Вибрати карту в блоці вибору карти;	У блоці обраної карти з'явилася інформація про неї. У блоці історії транзакцій з'явилися всі записи історій по обраній карті.	• пройдений

Таблиця 4.5 – Тест-кейс№5

Назва:	Тест-кейс№5	
Функція:	Видалення карти	
Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Відкрити додаткову сторінку в додатку; 2. У блоці видалення карти вибрати карту і натиснути на кнопку закрити карту;	З'явилося попередження що кошти на картці перейдуть до фонду банку і кнопка підтвердження	• пройдений
Кроки тесту:		
3. Натиснути на кнопку підтвердити;	Кошти перевелися до фонду банку, карта і її історія видалені	• пройдений

Таблиця 4.6 – Тест-кейс№6

Назва:	Тест-кейс№6	
Функція:	Поповнення картки через додаток терміналу	
Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Перейти на головну сторінку терміналу і ввести номер карти і пін-код;	Відкрилася сторінка терміналу з доступними функціями	• провальний

Продовження таблиці 4.6

Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Кроки тесту:		
2. Внести в купюроприймач необхідну суму; 3. Після внесення необхідної суми натиснути на кнопку поповнити;	Комісія списана і переведена до фонду банку, інші гроші надійшли на карту	• пройдений

Таблица 4.7 – Тест-кейс№7

Назва:	Тест-кейс№7	
Функція:	Вхід в обліковий запис	
Дія	Очікуваний результат	Результат теста: • пройдений • провальний • заблокований
Передумова:		
1. Відкрити головну сторінку в додатку;	Головна сторінка відкрита	• пройдений
Кроки тесту:		
2. Ввести email і пароль і натиснути на кнопку Увійти;	Якщо пароль і логін вірні то користувач переходить в обліковий запис	• пройдений

4.2 Виконання тестів

Виконання тестування на підставі тест-кейсів з розділу 4 представлено у вигляді баг-репорта (таблиця 4.8).

Таблиця 4.8 – Баг-репорт на тест-кейс №6

Короткий опис	Вхід не враховує пін-код введений користувачем.
Проект	Сайт банку
Компонент програми	Перевірка валідності пін-коду
Номер версії	1.0
Важливість: S1 Блокуючий (Blocker) S2 Критичний (Critical) S3 Значний (Major) S4 Незначний (Minor) S5 Тривіальний (Trivial)	S1 Значний (Major)
Пріоритет: P1 Високий (High) P2 Середній (Medium) P3 Низький (Low)	P1 Високий
Статус	Виправлений
Автор	Лебедев Я.С.
Призначений на	Лебедев Я.С.
Кроки відтворення	1. Перейти на головну сторінку терміналу і ввести номер карти і неправильний пін-код;
Фактичний Результат	Відкрита сторінка карти яка не належить користувачеві.

Продовження таблиця 4.8

Очікуваний результат	З'явився напис " пін код невірний", і користувачеві відмовлено в доступі.

5. ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ПРОДУКТУ

Додаток банку створювався для того що б поліпшити автоматизацію виконання необхідних операцій не знаходячись у банку фізично.

Для того що б відкрити додаток потрібно у браузері в рядку «адреса сайту» ввести адресу сайту. При першому вході в додаток користувач потрапляє на сторінку входу. Також на цій сторінці відображаються новини (рис. 5.1).

Рисунок 5.1 – Вхід до особистого кабінету

Якщо у користувача немає облікового запису те треба натиснути на кнопку "Створити аккаунт". Після цього користувач потрапляє на сторінку реєстрації (рисунок 5.2).

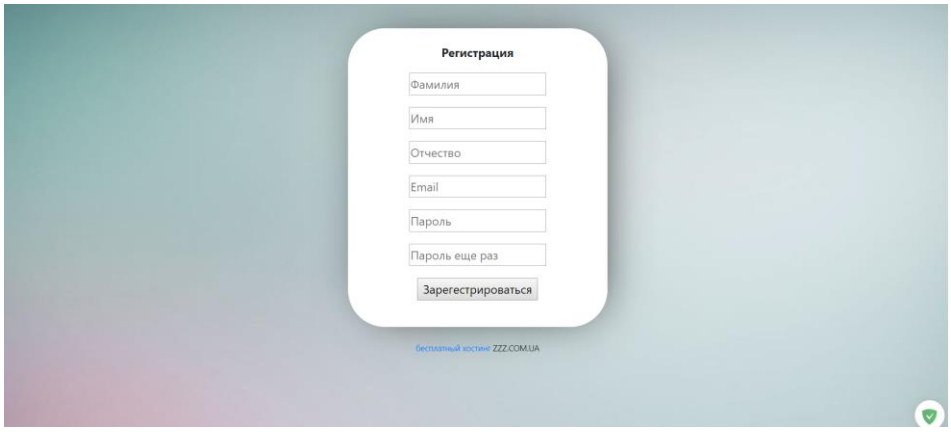

Рисунок 5.2 – реєстрація

Якщо у користувача вже є аккаунт те він може ввойти в аккаунт використовуючи email і пароль. Після входу у свій обліковий запис користувач потрапляє на сторінку з головними функціями (рис. 5.3).

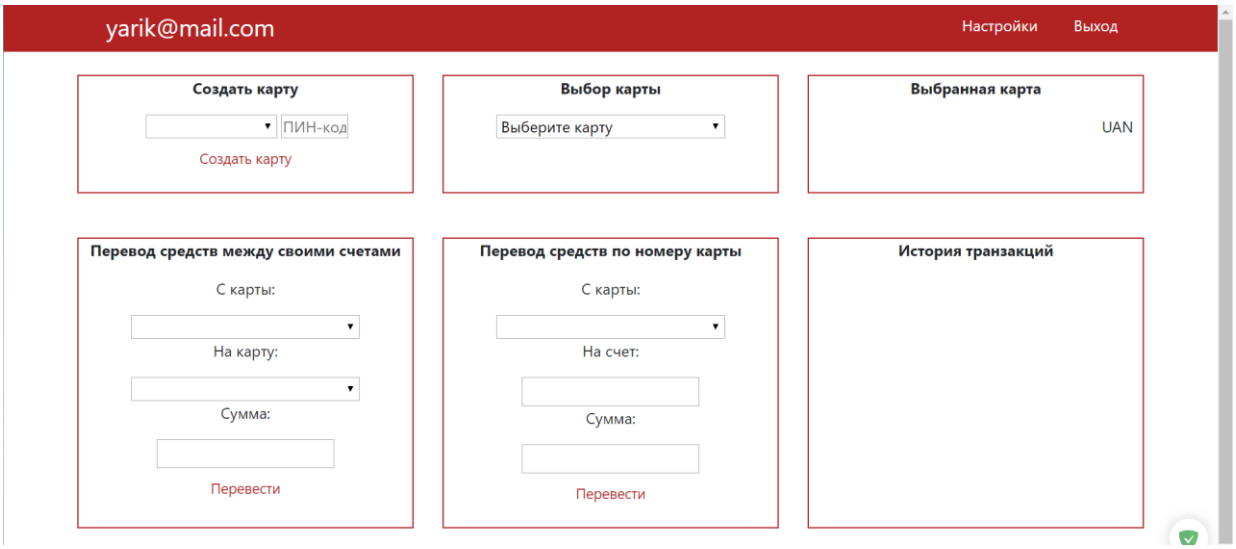

Рисунок 5.3 – Головна сторінка

Для створення нової карти користувача переходимо у блок "Створити карту" і вибираємо тип карти, після вводимо пін-код і натискаємо на кнопку створити карту після чого натискаємо на кнопку "Підтвердити" яка з'явилася ослові натиснення на кнопки "Створити карту" (рис. 5.4).

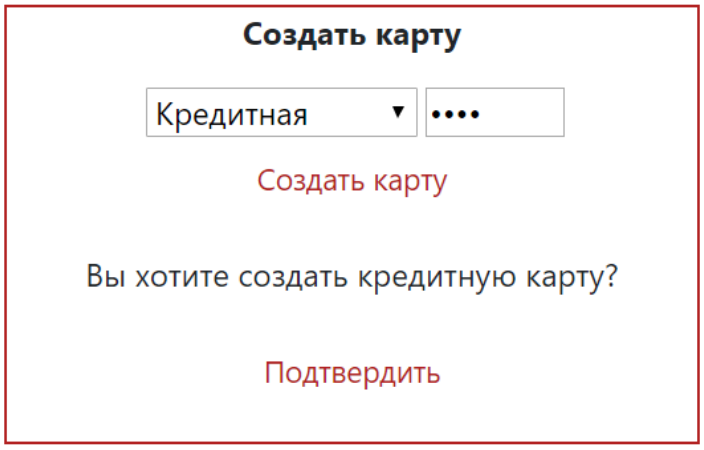

Рисунок 5.4 – Створення карти

Після того, як користувач підтвердить створення карти його повідомить про це напис з номером і типом карти (рис. 5.5).

Рисунок 5.5 – Повідомлення

Для того що б вибрати карту по якій користувач хоче подивитися інформацію треба перейти у блок "Вибір карти" і вибрати карти. Після натиснення на необхідну карту зі списку вона буде автоматично вибрана. У списку відображається номер карти і сума, карта підсвічуватиметься кольором який означає її тип (кредитна - червоний, для виплат - жовтий, накопичувальна - зелений) (рис. 5.6).

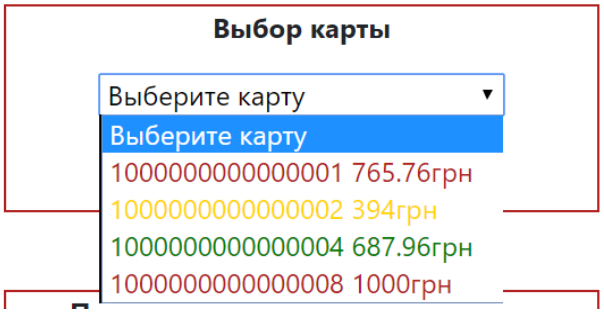

Рисунок 5.6 – Вибір карти

Після того, як користувач вибере карту те у блоці "Вибрана карта" з'явиться інформація про неї: номер карти, засоби на ній і тип карти (рис.5.7).

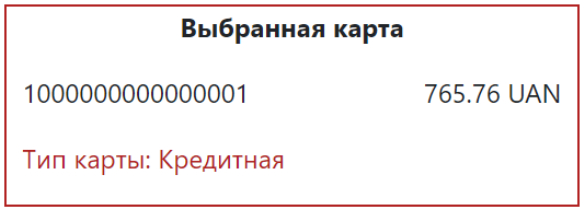

Рисунок 5.7 – Вибранна карта

Так само у блоці історії транзакцій з'явиться інформація про операції з цією картою (рис.5.8).

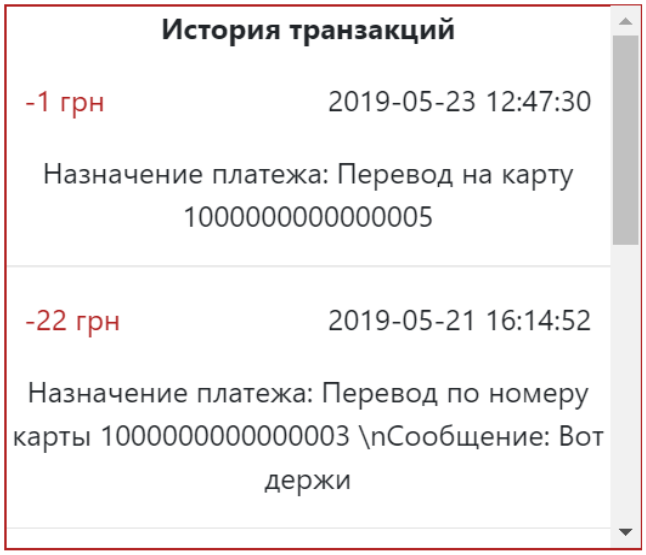

Рисунок 5.8 – історія транзакцій

Щоб перевести грошові кошти між своїми картами потрібно в блоці "переказ коштів між своїми рахунками" вибрати карту для переказу і карту на яку хочете перевести після потрібно ввести суму переказу, і натиснути на кнопку перевести (рис.5.9).

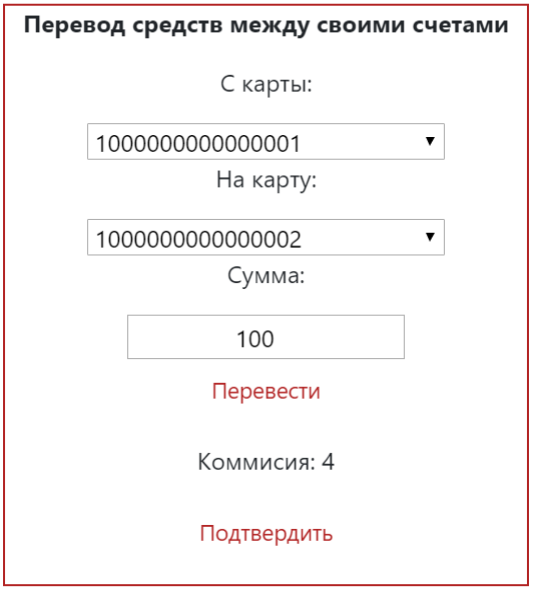

Рисунок 5.9 – Переведення між своїми картами

Після натискання на кнопку "Підтвердити" з'явиться повідомлення про переказ і в історіях транзакцій будуть записані нові дії, комісія піде до фонду банку.

Для переказу коштів за номером карти потрібно зробити те ж саме що і в попередньому блоці за винятком того що потрібно замість вибору карти на яку потрібно перевести кошти ввести номер карти (16 цифр) на яку хочете перевести (рис.5.10). Після натискання на кнопку "Перевести" з'явиться підтвердження переказу де буде написана комісія, також з'явиться поле для введення повідомлення "Призначення платежу". Після того, як користувач підтвердить операцію з'явиться повідомлення про операції і в історію транзакцій будуть записані нові дії з картою.

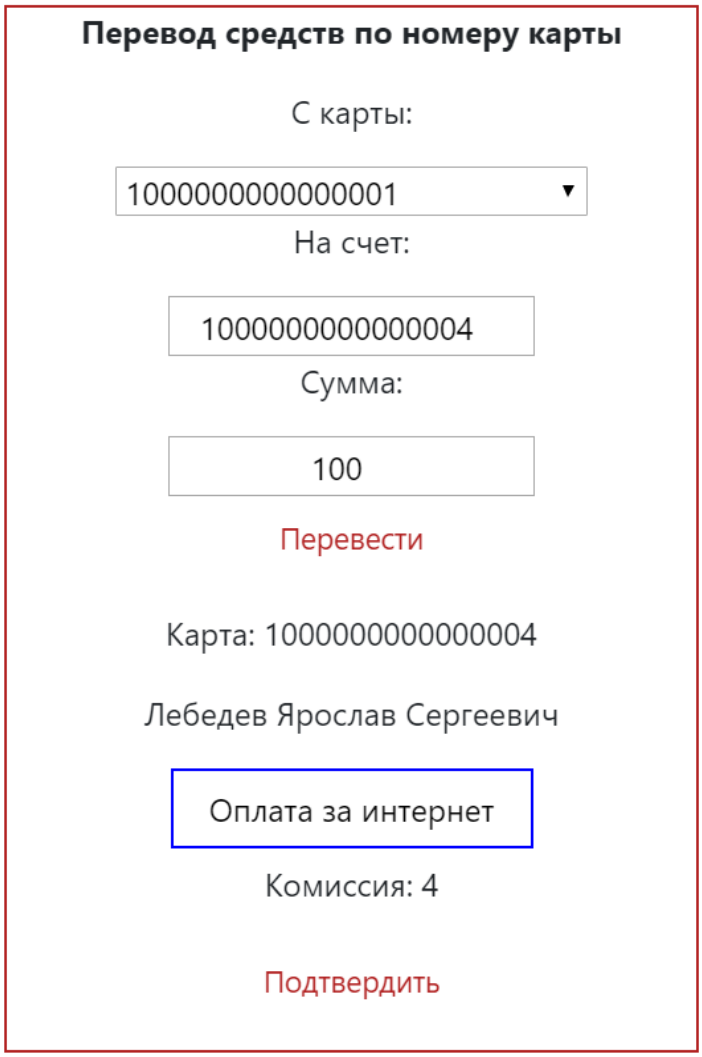

Рисунок 5.10 – Переведення по номеру карти

Якщо натиснути на кнопку "Налаштування" у верхній панелі сайту, користувач перейде на сторінку профілю де може переглянути свої дані а так само закрити карту (рис.5.11).

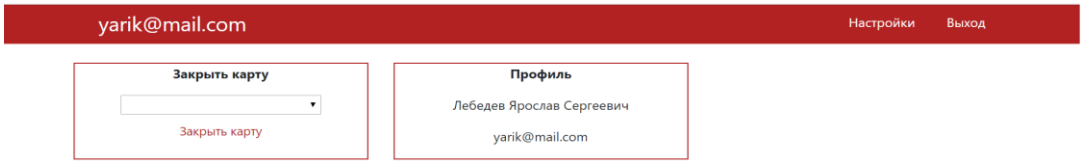

Рисунок 5.11 – Сторінка налаштувань

Щоб закрити карту потрібно вибрати її в блоці "Закрити карту". Для закриття картки вона повинна бути погашена, якщо це кредитна картка. Всі гроші з карти повинні бути виведені або переведені на іншу карту. Якщо на карту залишаться гроші то вони підуть у фонд банку. Після натискання на кнопку "Закрити карту" з'явиться підтвердження закриття де буде попередження про те що залишилися кошти піде до фонду банку. Після натискання на кнопку "Підтвердити" карта буде видалена (рис.5.12).

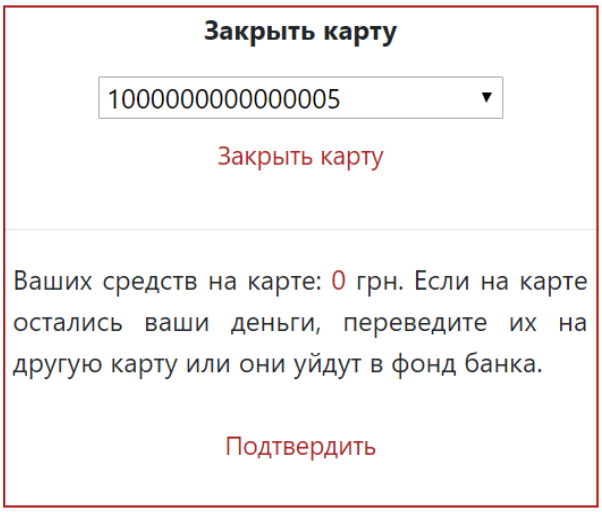

Рисунок 5.12 – Закриття карти

Щоб завершити роботу з сайтом потрібно натиснути на кнопку "Вихід у верхній панелі сайту.

Для роботи з терміналом користувачеві потрібно підійти до нього (вони розставлені в самих людних місцях, магазинах, торгових центрах, а так само у відділеннях банку). Після того як користувач підійде до нього він побачить поле де потрібно ввести номер карти (16 цифр) і пін-код. Після того як він натисне увійти в термінал він потрапить в головне вікно (рисунок 5.13).

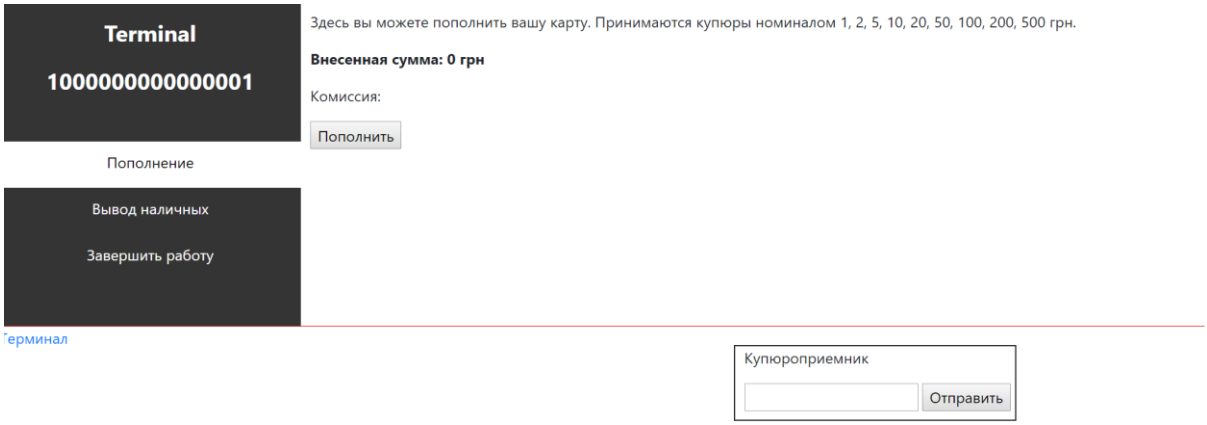

Рисунок 5.13 – Термінал «Поповнення»

Для поповнення картки користувачу потрібно перейти в розділ поповнення і ввести потрібну суму готівки в купюроприймач. Після користувачеві покажуть суму комісії. Коли він натисне на кнопку "Поповнити" карта буде поповнена.

Для виведення готівки користувачеві потрібно перейти в розділ виведення готівки і вибрати суму яку він хоче вивести. Після того як він натисне на кнопку "вивести готівку" термінал видасть користувачеві готівку, комісія буде списана з карти (рисунок 5.14).

Для завершення роботи з терміналом потрібно натиснути на кнопку "Завершити роботу".

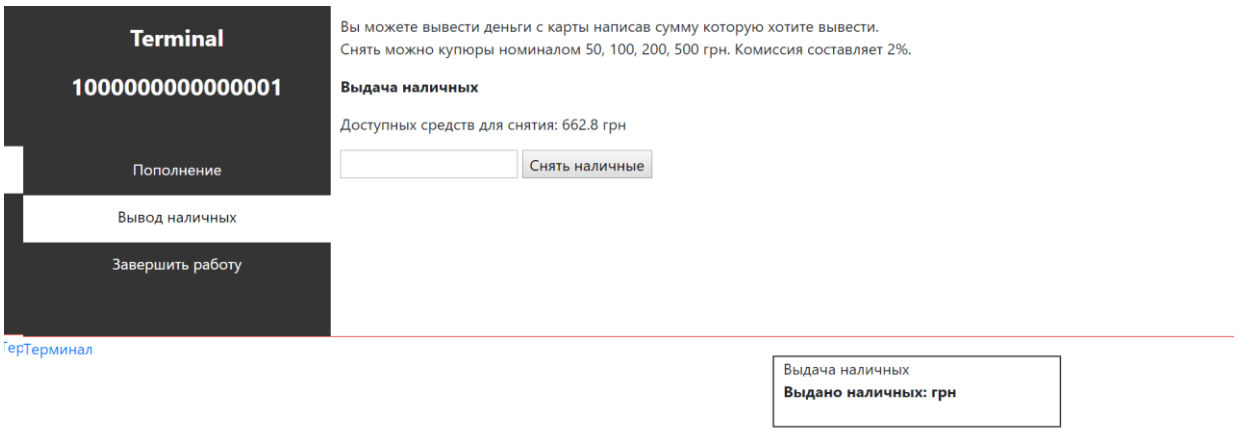

Рисунок 5.14 – Зняття готівки

6. ЕКОНОМІЧНЕ ПОЯСНЕННЯ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

6.1. Визначення трудомісткості розробки програмного забезпечення

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}, \text{ людино-годин} \qquad (6.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі;

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

$$t = 73 + 212 + 96 + 483 + 26 + 108 = 998, \text{ людино-годин}$$

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (6.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми, 0,9-1,2;

p – коефіцієнт кореляції програми в ході її розробки 0,05-0,1.

$$Q = 966 \cdot 1 \cdot (1 + 0{,}1) = 1062{,}6 \text{ людино-годин}$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} \text{ людино-годин,} \quad (6.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

Коефіцієнт збільшення витрат праці внаслідок недостатнього опису завдання B – якість постановки задачі, виданої для розробки програми, з зв'язку з тим, що завдання, як правило, вимагають уточнення і доопрацювання (практика показує, що в більшості випадків цей коефіцієнт в залежності від складності завдання приймається від 1,2 до 1,5).

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Коефіцієнт кваліфікації розробника k ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежно від стажу роботи і становить: для працюючих до двох років – 0,8; від двох до трьох років – 1,0; від трьох до п'яти років – 1,1-1,2; від п'яти до семи років – 1,3-1,4; понад сім років – 1,5-1,6).

$$t_u = \frac{1062{,}6 \cdot 1{,}2}{75 \cdot 0{,}8} = 21{,}25 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20K25) \cdot k} \text{ людино-годин} \quad (6.4)$$

$$t_a = \frac{1062,6}{20 \cdot 0,8} = 66,41 \text{ людино-годин}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20K25) \cdot k} \text{ людино-годин} \quad (6.5)$$

$$t_n = \frac{1062,6}{20 \cdot 0,8} = 66,41 \text{ людино-годин}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) \cdot k} \text{ людино-годин,} \quad (6.6)$$

$$t_{\text{отл}} = \frac{1062,6}{4 \cdot 0,8} = 332,06 \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}} \text{ людино-годин,} \quad (6.7)$$

де $t_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{Q}{(15..20) \cdot k} \text{ людино-годин} \quad (6.8)$$

$t_{\text{до}}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{\text{до}} = 0,75 \cdot t_{\text{др}} \text{ людино-годин} \quad (6.9)$$

$$t_{до} = 0,75 \cdot 88,55 = 66,41 \text{ людино-годин,}$$

$$t_{\text{др}} = \frac{1062,6}{15 \cdot 0,8} = 88,55 \text{ людино-годин}$$

$$t_{\text{до}} = 0,75 \cdot 88,55 = 66,41 \text{ людино-годин.}$$

$$t_{\text{д}} = 88,55 + 66,41 = 155 \text{ людино-годин.}$$

6.2. Витрати на створення програмного забезпечення

Тарифна система використовується для диференціації розмірів заробітної плати працівників залежно від їхньої кваліфікації, відповідальності, умов праці, її кількості та результатів. З допомогою тарифної системи установлюються співвідношення між низько і високооплачуваними категоріями трудящих. Тарифна система відображає поділ працівників за професіями, спеціальностями та кваліфікацією. Тарифна система є основою для установлення правильного співвідношення між темпами зростання продуктивності праці та середньої заробітної плати працівників. З її допомогою визначається необхідна кількість працівників відповідної кваліфікації чи спеціальності, а також співвідношення заробітної плати різних категорій працівників.

Форми і системи заробітної плати — це механізм встановлення розміру заробітку в залежності від кількості та якості праці і її результатів. Обираючи певну форму заробітної плати і конкретну систему формування заробітку, роботодавець управляє інтенсивністю та якістю праці конкретних працівників. Розрізняють дві основні форми заробітної плати: почасову і відрядну. При почасовій формі заробітної плати мірою праці виступає відпрацьований час, а заробіток працівнику нараховується згідно з його тарифною ставкою чи посадовим окладом за фактично відпрацьований час. При відрядній формі заробітної плати мірою праці є вироблена працівником продукція (або виконаний обсяг робіт), а розмір заробітку прямо пропорційно залежить від її

кількості та якості, виходячи із встановленої відрядної розцінки. Системи заробітної плати характеризують взаємозв'язок елементів заробітної плати: тарифної частини, доплат, надбавок, премій. Зрозуміло, що варіантів такого взаємозв'язку може бути безліч, і будь-який з них, реально існуючий на конкретному підприємстві, є системою заробітної плати. В країнах ринкової економіки системи заробітної плати, що використовуються на підприємствах, розглядаються як ноу-хау і не розголошуються.

Види систем заробітної плати:

– Пряма відрядна система оплати праці. За індивідуальної прямої відрядної системи розмір заробітної плати прямо залежить від результатів роботи кожного робітника.

– Підрядно-преміальна система оплати праці. Сутність її полягає в тому, що загальний заробіток робітника складається з заробітної плати, нарахованої за фактично виконану роботу чи вироблену продукцію за прямими відрядними розцінками, і премії за виконання та перевиконання установлених планових кількісних і якісних показників.

– Відрядна-прогресивна система. Сутність її полягає в тому, що заробітна плата робітникам нараховується за обсяг виконаної роботи чи виробленої продукції в межах планової норми виробітку за основними незмінними відрядними розцінками, а за обсяг роботи чи продукції понад вихідну планову норму, – за підвищеними чи прогресивно зростаючими прямими відрядними розцінками.

– Непряма відрядна система заробітної плати, використовується для визначення заробітку допоміжних робітників.

– Акордна система оплати праці. Застосовується для окремих груп робітників. Сутність її полягає у тому, що відрядна розцінка установлюється не на окрему виробничу операцію, а на весь комплекс робіт загалом, виходячи із діючих норм часу і розцінок.

– Погодинна форма оплати праці. За погодинної форми заробітної плати оплата праці здійснюється за годинними тарифними ставками з

урахуванням відпрацьованого часу та рівня кваліфікації, що визначається тарифним розрядом.

Доплати та надбавки – це самостійний елемент заробітної плати з погляду її структури. Водночас вони є складовою тарифної системи. Остання в класичному розумінні є інструментом диференціації та регулювання рівня заробітної плати різних груп і категорій працівників залежно від кваліфікаційного рівня, складності виконуваних робіт, їх відповідальності, а також умов та інтенсивності праці, специфічних особливостей підриємства. З огляду на функціональне призначення доплати та надбавки є тим елементом тарифної системи, за допомогою якого компенсують суттєві відхилення від умов роботи, які визнаються нормальними й безпосередньо не враховуються в тарифних ставках і посадових окладах.

За функціональним призначенням всі види надбавок і доплат можна поділити на стимули, які пов’язані:

– З кількісними показниками праці (тобто ті, які стимулюють або компенсують збільшення в певному обсязі робіт, які виконуються працівниками, зокрема, за розширення зони обслуговування, виконання додаткових функцій тощо);

– З якісними показниками роботи(ті, які здійснюють додаткове стимулювання підвищення кваліфікації, професійної майстерності, високих стійких результатів праці);

– З кількісно-якісними показниками праці (ті, які стимулюють виконання в строки і якісно роботи, складність, напруженість і високу якість роботи).

Розрахунок годинної оплати програміста

Розрахуємо середню годинну оплату програміста. Для цього необхідно спочатку визначити його річний фонд грошового забезпечення. Це можна зробити, знаючи місячне грошове забезпечення програміста. Воно складає приблизно 27629 гривень. Таким чином, річний фонд грошового забезпечення

331548 гривень. Визначимо число робочих годин у році за наступною формулою:

$$n(p) = (N - N(n) - N(\text{в})) \cdot 8, \quad (6.10)$$

де N – загальне число днів у році,

$N(n)$ – число святкових днів у році,

$N(\text{в})$ – число вихідних днів у році.

Число святкових днів у році – 11, а вихідних – 126. Отже, число робочих годин у році дорівнює:

$$n(p) = (365 - 11 - 126)) \cdot 8 = 1824(\text{год}).$$

Середня годинна оплата програміста визначається співвідношенням:

$$\text{С}_{\text{розр.}} = \frac{\text{ФЗР}_{\text{СН}}}{n(p)}, \quad (6.11)$$

де $\text{ФЗР}_{\text{сн}}$ – річний фонд грошового забезпечення.

$$\text{С}_{\text{розр.}} = \frac{331548}{1824} = 181{,}76(\text{грн}).$$

Отже, витрати по основній оплаті праці ($\text{З}_{\text{осн.}}$) розробників програми складають:

$$\text{З}_{\text{осн.}} = 966 \cdot 181{,}76 = 175580{,}16(\text{грн}).$$

Для визначення загальної суми витрат на оплату праці необхідно додати доплати та надбавки . Приймаємо розмір додаткової заробітної плати 15% від основної заробітної плати:

$$З_{доп.} = 175580{,}16 \cdot 0{,}15 = 26337(грн).$$

Загальний фонд оплати праці:

$$З_{заг.} = 175580{,}16 + 26337 = 201917{,}16(грн).$$

Визничим розмір єдиного соціального внеску, який складає 22%:

$$ЄСВ = 201917{,}16 \cdot 0{,}22 = 44421{,}77(грн).$$

$$З_{розр} = 201917{,}16 + 44421{,}77 = 246338{,}93(грн).$$

Визначимо витрати, пов'язані з розробкою програми на ПК. Вони розраховуються як добуток часу використання ПК для розробки програми на собівартість машинного часу обчислювальної техніки.

Собівартість ($С_{ПК}$) однієї години роботи ПК дорівнює відношенню річних поточних витрат на експлуатацію ПК ($З_{гПК}$) до річного фонду часу ($Т_{гПК}$) корисної роботи ПК.

$$С_{ПК} = З_{гПК}/Т_{гПК} \qquad (6.12)$$

Розрахуємо річний фонд часу роботи ПК.

Визначивши дійсний річний фонд часу ЕОМ у годинах, одержимо можливість оцінити собівартість годин машинного часу. Дійсний річний фонд часу ЕОМ дорівнює числу робочих годин у році для оператора, за винятком часу на профілактику й ремонт ЕОМ. Час профілактики: щомісячно – 5 годин; щорічно – 6 діб.

$$Т_{гПК} = 1824 - (6 \cdot 8 + 5 \cdot 12) = 1716(год)).$$

Річні поточні витрати на експлуатацію визначаються по формулі:

$$З_{гПК} = З_{гАМ} + З_{гЕЛ} + З_{гРЕМ} + З_{гМАТ} + З_{гДР}, \quad (6.13)$$

де $З_{гАМ}$ – річні відрахування на амортизацію,

$З_{гЭЛ}$ – річні витрати на електроенергію для ПК,

$З_{гРЕМ}$ – річні витрати на ремонт ПК,

$З_{гМАТ}$ – річні витрати на додаткові комплектуючі ПК,

$З_{гДР}$ – інші витрати.

Сума річних амортизаційних відрахувань визначається за наступною формулою:

$$З_{гАМ} = Ц_{ПК} + НА, \quad (6.14)$$

де $Ц_{ПК}$ – балансова вартість ПК,

НА – норма амортизаційних відрахувань, дорівнює 15% (у квартал).

Балансова вартість ПК:

$$Ц_{ПК} = Ц_{р} \cdot (1 + К_{ун}), \quad (6.15)$$

де $Ц_{р}$ – ринкова вартість ПК,

$К_{ун}$ – коефіцієнт, що враховує витрати на установку й налагодження, рівний 12%.

$$Ц_{ПК} = 26000 \cdot (1 + 0{,}12) = 25760 \text{ (грн)}$$
$$З_{тАМ} = 25760 \cdot 0{,}15 \cdot 4 = 15456{,}6 \text{ (грн)}.$$

Витрати на електроенергію, споживану ПК, визначаються за наступною формулою:

$$З_{гЕЛ} = Р_{чПК} \cdot Т_{гПК} \cdot Ц_{ЕЛ} \cdot А, \quad (6.16)$$

де $Р_{чПК}$ – настановна потужність ПК ($Р_{чПК}$=0,3 (кВт)),

$Т_{гПК}$ – річний фонд корисного часу роботи машини,

$Ц_{ЭЛ}$ – вартість 1 квт/година електроенергії ($Ц_{ЭЛ}$=1,5 (грн)),

А – коефіцієнт інтенсивного використання ПК (0,9-1).

Таким чином, розрахункове значення витрат на електроенергію, споживану ПК, складає:

$$З_{гЕЛ} = 0{,}4 \cdot 1716 \cdot 0{,}93 \cdot 1 = 638{,}35(грн).$$

Витрати на поточний і профілактичний ремонт приймаються рівними 6% від вартості ПК:

$$З_{гРЕМ} = Ц_{ПК} \cdot 0{,}06 \quad (6.17)$$

$$З_{гРЕМ} = 25760 \cdot 0{,}06 = 1545{,}6(грн)$$

Витрати на матеріали – витрати необхідні для забезпечення експлуатації ПК приймаються рівними 2% від вартості ПК.

$$З_{гМАТ} = Ц_{ПК} \cdot 2\% \quad (6.18)$$

$$З_{гМАТ} = 25760 \cdot 0{,}02 = 5151{,}2(грн).$$

Непрямі витрати, тобто витрати пов'язані з експлуатацією ПК приймаються рівними 5-10% вартості ПК.

$$З_{гДР} = Ц_{ПК} \cdot 5\% \quad (6.19)$$

$$З_{гДР} = 25760 \cdot 0{,}05 = 1288(грн).$$

Повні витрати на експлуатацію ПК впродовж року складають:

$$З_{гПК} = 15456{,}6 + 638{,}35 + 1545{,}6 + 5151{,}2 + 1288 = 24079{,}75(грн).$$

Собівартість машинного часу ($С_{ПК}$) складає:

$$С_{ПК} = 24079{,}75/1716 = 14{,}04(грн).$$

У ході розробки програмного комплексу машина використовувалася на етапах програмування:

- написання програми за готовою схемою алгоритму;
- налагодження програми на ПК;
- підготовки документації по задачі.

Таким чином, витрати машинного часу склали ($t_{маш}$):

$$t_{маш} = t_{прог} + t_{отл} + t_{док}. \quad (6.20)$$

$$t_{маш} = 66{,}41 + 26 + 155 = 247{,}41(чол./год)$$

Витрати на оплату машинного часу можна розрахувати по формулі:

$$З_{маш} = t_{маш} \cdot С_{ПК}. \quad (6.21)$$

$$З_{маш} = 247{,}41 \cdot 14{,}04 = 3473{,}63(грн).$$

Загальні витрати на створення програмного продукту складають:

$$З_{разом} = З_{разр} + З_{маш}. \quad (6.22)$$

$$З_{разом} = 246338{,}93 + 3473{,}63 = 249812{,}56(грн).$$

Таблиця 6.1 – Кошторис витрат на створення програмного продукту

Стаття витрат		Сума, грн.	В відсотках аід загальної суми
ФЗП	$З_{осн}$	175580,16	64,69
	$З_{доп}$	26337	9,7
ЄСВ		44421,77	16,37
Експлуатаційні витрати	$З_{гАМ}$	15456,6	5,69
	$З_{гЭЛ}$	1638,35	0,6
	$З_{гРЕМ}$	1545,6	0,57
	$З_{гМАТ}$	5151,2	1,9
	$З_{гДР}$	1288	0,47
Матеріали і комплектуючи			
Всього		271418,68	100

Визначимо відпускну ціну програми:

- прибуток від реалізації 15%;
- податок на додану вартість 20%

$$Ц_{п} = З_{разом} + П_{р} \quad (6.23)$$

$$Ц_{п} = 249812{,}56 \cdot 1{,}15 = 287284{,}44(грн).$$

Вартість програми з урахуванням ПДВ:

$$Ц_{В} = Ц_{п} + ПДВ \quad (6.24)$$

$$Ц_В = 287284,44 \cdot 1,2 = 344741,32(грн.)$$

6.3. Економічне обґрунтування

Банки є основним фінансовим посередником в економіці. Дипломна робота націлена покращити автономність і швидкість взаємодії банку з користувачами. Веб додаток дозволить користувачам використовувати банк де б вони не були і з яким би пристроєм вони не користувалися, при цьому не турбуючись про безпеку їх дій і збереження їх коштів. Якщо додаток дозволить користувачам зручно і продуктивно працювати з банком, не перебуваючи в ньому, то це без сумніву стане для банку набагато більш прибутково ніж, коли не було банківської системи. Крім цього, тепер, коли є автоматизована система банку не обов'язково наймати додаткових співробітників, які робили те що зараз може робити комп'ютер. Так само користувачам не потрібно буде стояти в великих чергах для того що б подивитися виписку по їх карті.

Саме зараз банк в середньому обслуговує за день 3000 клієнтів, при цьому працівникам відділень Банку, яких складає 30 співробітників, відраховується заробітна плата в розмірі 420000 грн. за місяць, при використанні додатка банк за ту ж саму заробітну плату буде обслуговувати в середньому 20000 клієнтів, отже обсяг роботи розрахуємо як співвідношення кількості людей, які були обслужені за день:

$$О_{роботи} = (К_{з.сис.} - К_{без.сис.})/К_{з.сис.}, \quad (6.25)$$

де $О_{роботи}$ – об’єм роботи,

$К_{без.сис}$ – кількість відвідувачів бібліотеки, які були обслужені без системи,

$К_{з.сис.}$ – кількість відвідувачів бібліотеки, які були обслужені з системою.

$$О_{роботи} = \frac{20000-3000}{20000} = 0{,}85 = 85\%,$$

Отже об'єм роботи, за допомогою програми, буде збільшено на 85%, враховуючи заробітну платню банку знайдемо умовний економічний ефект (УЕ) від програми:

$$УЕ_{місяць} = О_{роботи} \cdot Зароб.\,плата. \quad (26)$$

$$УЕ_{місяць} = 0{,}85 \cdot 420000 = 357000(грн).$$

$$УЕ_{рік} = З_{місяць} \cdot 12, \quad (6.27)$$

$$УЕ_{рік} = 357000 \cdot 12 = 4284000(грн).$$

Після того коли знайшли умовний ефект, тобто оцінивши труд банку в гривневому виразі знайдемо через який час програма себе окупить:

$$О_{програми} = З_{разом}/УЕ_{рік}, \quad (6.28)$$

де $О_{програми}$ – час за який програма себе окупить.

$$О_{програми} = \frac{249812{,}56}{4284000} = 3 \text{ нед } 4 \text{ дні}. \quad (29)$$

Отже ми знайшли, що програма окупиться за 3 тижня 4 дні, після чого округлюємо наш час до ближнього більшого цілого числа, тобто отримуємо 4 тижня, і вже після цього розраховуємо ефект за 4 тижня:

$$Е_{n\ роки} = n \cdot УЕ_{рік}, \quad (6.30)$$

де $Е_{\text{n роки}}$ – ефективність за *n* років.

$$Е_{4\ \text{нед}} = 4 \cdot 89250 = 357000(\text{грн}).$$

Прибуток за дві неділі буде складати:

$$П_{\text{n нед}} = Е_{\text{n нед}} - З_{\text{разом}}, \quad (6.31)$$

де $П_{\text{n нед}}$ – прибуток за n неділь.

$$П_{\text{4 нед}} = 357000 – 249812,56 = 107187,44(\text{грн}).$$

Ефективність за дві неділі буде складати:

$$Еф_{n\ \text{роки}} = \%, \quad (6.32)$$

де $Е_{\text{n роки}}$ – ефективність за *n* років.

$$Еф_{\text{4 нед}} = (107187,44 / 249812,56) * 100 = 43\% .$$

Отже можна зробити висновок, що програма буде окуплена за 4 тижня. За чотири тижня її експлуатації вона принесе прибуток в розмірі 107187,44 гривень, а економічна ефективність програми за чотири тижня складе 43%.

7. ОХОРОНА ПРАЦІ

Охорона праці – система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних і лікувально-профілактичних заходів та засобів, що забезпечують безпеку, збереження здоров'я і працездатності людини в процесі праці. Науково-технічний прогрес в ніс серйозні зміни в умови виробничої діяльності працівників розумової праці. Їх праця стала більш інтенсивним, напруженим, які вимагають значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці і відпочинку.

Охорона здоров'я трудящих, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства. Звертається увага на необхідність широкого застосування прогресивних форм наукової організації праці, зведення до мінімуму ручної, малокваліфікованої праці, створення обстановки, що виключає професійні захворювання і виробничий травматизм.

Даний розділ дипломної роботи присвячений розгляду наступних питань:

– організація робочого місця програміста;
– визначення оптимальних умов праці програміста.

7.1 Аналіз умов праці на робочому місті

Організація робочого місця

Аналіз умов праці показує, що у приміщенні в офісі на програміста можуть негативно впливати наступні фізичні та психофізіологічні фактори:

1. підвищена або знижена температура повітря робочої зони;
2. підвищена або знижена вологість повітря;
3. недостатня освітленість робочого місця;

4. підвищений рівень шуму на робочому місці;

5. підвищений рівень електромагнітних випромінюване;

6. нервово-психічні перевантаження (розумова перенапруга, перенапруга аналізаторів);

7. фізичні перевантаження (одноманітна поза викликає статичну втому).

Основним робочим положенням є положення сидячи. Робоче місце для виконання робіт в положенні сидячи зорганізується відповідно до ГОСТ 12.2.032-78.

Робоча поза сидячи викликає мінімальне стомлення програміста. Раціональне планування робочого місця передбачає чіткий порядок і сталість розміщення предметів, засобів праці і документації. Те, що потрібно для виконання робіт частіше, розташоване в зоні легкої досяжності робочого простору.

Моторне поле – простір робочого місця, в якому можуть здійснюватися рухові дії людини.

Максимальна зона досяжності рук – це частина моторного поля робочого місця, обмеженого дугами, що описуються максимально витягнутими руками при русі їх в плечовому суглобі.

Оптимальна зона – частина моторного поля робочого місця, обмеженого дугами, описуваними передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем.

Важливим елементом робочого місця програміста є крісло. Воно виконується відповідно до ДСТУ 8604:2015. При проектуванні крісла виходять з того, що при будь-якому робочому положенні програміста його поза повинна бути фізіологічно правильно обґрунтованої, тобто положення частин тіла повинно бути оптимальним. Для задоволення вимог фізіології, що випливають з аналізу положення тіла людини в положенні сидячи, конструкція робочого сидіння повинна відповідати таким основним вимогам:

– допускати можливість зміни положення тіла, тобто забезпечувати вільне переміщення корпусу і кінцівок тіла один щодо одного;

– допускати регулювання висоти в залежності від росту працюючої людини (у межах від 400 до 550 мм);

– мати злегка увігнуту поверхню,

– мати невеликий нахил назад (5°).

Важливим моментом є також раціональне розміщення на робочому місці документації, канцелярських принадлежностей, що повинно забезпечити працюючому зручну робочу позу, найбільш економічні руху і мінімальні траєкторії переміщення працюючого і предмета праці на даному робочому місці.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинна сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, в яких виконується одноманітна розумова робота, що потребує значної нервової напруги і великого зосередження, фарбування повинна бути спокійних тонів – мало насичені відтінки холодного зеленого або блакитного кольорів.

Оптимальна зона – частина моторного поля робочого місця, обмеженого дугами, описуваними передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем представлено на рисунку 7.1.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість, шум і мікроклімат.

Мікроклімат робочої зони програміста

Мікроклімат робочої зони – це клімат внутрішнього середовища виробничих помешкання, що визначаються сполучення температури повітря, відносної вологості повітря, швидкості прямування повітря, барометричного тиску та інтенсивності теплового випромінювання.

Оптимальні мікрокліматичні умови – це поєднання параметрів мікроклімату, які при тривалому та систематичного впливів на людину забезпечують зберігання нормального теплового стану організму без активації механізмів теплорегуляції.

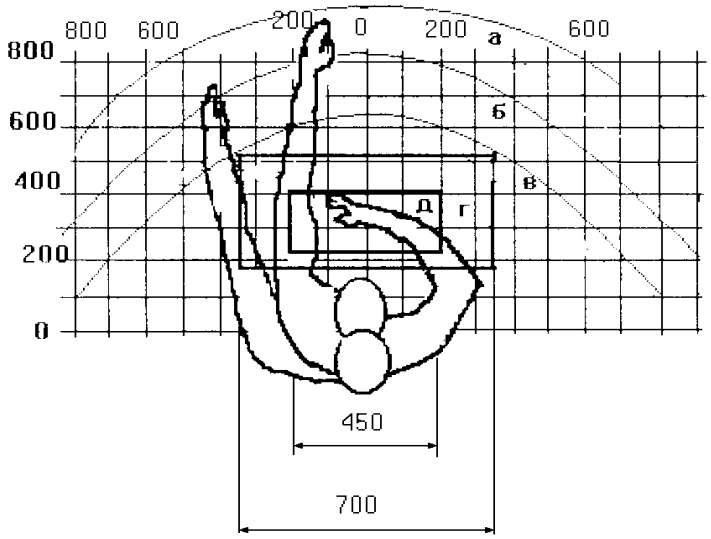

Рисунок 7.1 – Оптимальна зона

Допустимі мікрокліматичні умови – це поєднання параметрів мікроклімату, які при тривалому та систематичного впливів на людину можуть викликати зміни теплового стану організму, що швидко минають и нормалізуються та супроводжуються напруженого механізмів терморегуляції в межах фізіологічної норми. При цьому не виникає пошкодженню або порушеннях стану здоров'я, но можуть спостерігатись дискомфорт, погіршення самопочуття та зниження працездатності.

Норми мікроклімату встановлюються в залежності від пори року, характеру трудового процесах и характером виробничого приміщення (табл. 7.1).

Таблиця 7.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютери.

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22 ... 24 ° С 40 ... 60% до 0,1 м / с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	23 ... 25 ° С 40 ... 60% 0,1 ... 0,2 м / с

Освітлення робочого місця

Правильно спроектоване и виконання виробниче освітлення покращує умови зорової роботи, зніжує стомлюваність, сприяє підвищенню продуктивності праці, благотворно впливає на виробниче середовище, надаючи позитивну психологічну дію на працюючий, підвищує безпеку праці и зніжує травматизм.

Існує три види освітлення – природнє, штучне и суміщене (природне і штучне разом), при якому недостатнє за нормами природне доповнюється штучним.

Природнє освітлення – освітлення приміщень деннім світлом, що потрапляє через Світлові Прорізи в зовнішніх огороджуючи конструкціях приміщення. Природньо освітлення характеризується тім, що змінюється в широких межах залежних від часу дня, пори року, характеру області і ряду других чинників.

Штучне освітлення застосовується при рабаті в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловій день). Освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним, називається змішанім освітленням.

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, у свою чергу, може бути загальним або комбінованим. Загальне - освітлення, при якому світильники розміщуються у

верхній зоні приміщення рівномірно, або, як розташоване устаткування. Комбіноване – освітлення, при якому до загально додається місцеве освітлення.

Вплив шуму на програміста

Робоче місце програміста устатковане монітором, вінчестером в системному блоці, трьома вентиляторами системи охолоджування персонального комп’ютера та клавіатурою. Крім того поряд працює периферійна техніка. Таким чином у приміщенні мають місце шуми механічного і аеродинамічного походження, широкосмугові із аперіодичним підсиленням при роботі принтерів. Орієнтовні еквівалентні рівні звукового тиску джерел шуму, що діють на програміста на його робочому місці, представлені в таблиці 7.2. Допустимий еквівалентний рівень шуму для робочого місця програміста складає 50 дБА.

Таблиця 7.2 – Рівні звукового тиску від різних джерел.

Джерело шуму	Рівень шуму, дБА
Принтер	45
Вентилятор	45
Жорсткий диск	55
Мишка	20
Клавіатура	25

Рівень шуму перевищує гранично допустимий рівень шуму для робочого місця програміста, тобто слід передбачити заходи по зниженню рівня шуму.

Встановлено, що шум погіршує умови праці, роблячи шкідливий вплив на організм людини. При тривалому впливі шуму на людину відбуваються небажані явища: знижується гострота зору, слуху, підвищується кров'яний тиск, знижується увага. Сильний тривалий шум може стати причиною функціональних змін серцево-судинної і нервової систем.

Виробничі випромінювання

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп'ютера представлені в таблиці 7.2. Нормованим параметром невикористаного рентгенівського випромінювання виступає потужність експозиційної дози. На відстані 5 см від поверхні екрану монітору її рівень не повинен перевищувати 100 мкР/год. Максимальний рівень рентгенівського випромінювання на робочому місці програміста зазвичай не перевищує 20 мкР/год.

Монітори є джерелом оптичного випромінювання. Найбільшу біологічну активність має активна зона ультра-фіолетові випромінювання (довжина хвилі від 200 до 315нм).

На відстані 5-10 см від екрана і корпусу монітора рівні напруженості можуть досягати 140 В/м по електричній складовій, що значно перевищує допустимі значення.

Небезпека ураження електричним струмом

На робочому місці програміста з всього устаткування металевим є лише корпус системного блоку комп'ютера, але тут використовуються системні блоки, що відповідають стандарту фірми IBM, у яких крім робочої ізоляції передбачений елемент для заземлення і провід з жилою, що заземлює, для приєднання до джерела живлення.

Основні причини ураження людини електричним струмом на робочому місці:

1. дотик до металевих неструмоведучих частин (корпусу, периферії комп'ютера), що можуть виявитися під напругою в результаті ушкодження ізоляції;
2. перевантаження в мережі;
3. нерегламентоване використання електричних приладів;
4. відсутність інструктажу співробітників з правил електробезпеки.

На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кВ/м, тобто в 10 разів перевищує норму 20 кВ/м.

7.2 Розробка заходів з охорони праці

Нормалізація повітря робочої зони

Для створення й автоматичної підтримки в приміщенні незалежно від зовнішніх умов оптимальних значень температури, вологості, чистоти і швидкості руху повітря, у холодний час року використовується водяне опалення, у теплий час року застосовується кондиціонування повітря.

Виробниче освітлення

Під час аналізу освітлення на робочому місті програміста було встановлено, що воно не відповідає встановленим нормам, тому для покращення умов праці рекомендуємо збільшити рівень загальної освітленості приміщення шляхом встановлення додаткових ламп.

Також для підтримки запроектованого освітлення у чистому виді необхідно скласти графік, де передбачити очищення віконних блоків і світильників не менше 2 разів на рік. Для забезпечення, відносно постійності природного освітлення необхідно вікна обладнати сонце-захисними, регульованими жалюзі з коефіцієнтом відбиття 0,5-0,7.

Захист від виробничого шуму

На робочому місці програміста джерелами шуму, як правило, є технічні засоби, як то – комп'ютер, принтер, вентиляційне обладнання, а також зовнішній шум. Вони видають досить незначний шум, тому в приміщенні досить використовувати звукопоглинання. Зменшення шуму, що проникає в приміщення ззовні, досягається ущільненням по периметру притворів вікон і дверей. Під звукопоглинанням розуміють властивість акустично оброблених поверхонь зменшувати інтенсивність відбитих ними хвиль за рахунок перетворення звукової енергії в теплову. Звукопоглинання є досить

ефективним заходом щодо зменшення шуму. Найбільш вираженими звукопоглинальними властивостями володіють волокнисто-пористі матеріали: фібролітові плити, скловолокно, мінеральна вата, поліуретановий поропласт, пористий полівінілхлорид і ін. До звукопоглинальним матеріалів відносяться лише ті, коефіцієнт звукопоглинання яких не нижче 0.2.

Звуковбирні облицювання з зазначених матеріалів (наприклад, мати із супертонкого скловолокна з оболонкою зі склотканини потрібно розмістити на стелі і верхніх частинах стін). Максимальне звукопоглинання буде досягнуто при облицюванні не менше 60% загальної площі огороджувальних поверхонь приміщення.

Для зменшення шуму в аналізованій приміщення пропоную використовувати замість матричного принтера, що створює багато шуму, більш тихий – лазерний принтер.

Захист від електромагнітних полів

Для попередження впровадження небезпечної техніки всі дисплеї повинні бути сертифіковані.

Основними напрямами в процесі розробки засобів захисту від дії ЕМП радіочастот є: зменшення інтенсивності опромінювання безпосередньо від самого джерела, екранування джерела опромінювання, застосування засобів індивідуального захисту.

Засоби захисту мають відповідати таким вимогам: не викривляти істотно електромагнітне поле; не знижувати якості технічного обслуговування і ремонту; не знижувати продуктивності праці.

Основними заходами та засобами боротьби з шумом є:

1. зниження рівнів шуму в джерелі шуму в джерелі його утворили (при проектуванні);
2. використання звуко поглинаючих та звукоізолюючих засобів;
3. раціональне планування виробничих приміщень та робочих місць.

Електробезпека

Електробезпечність у приміщенні лабораторії пропоную забезпечити

наступними технічними способами і засобами захисту:

1 для зменшення накопичення статичної електрики застосовувати зволожувачі і нейтралізатори, антистатичне покриття підлоги;

2 забезпечити приєднання металевих корпусів устаткування до жили, що заземлює, заземлення корпусу персонального комп'ютера забезпечити розетками з заземленням.

3 також організаційними заходами:

4 своєчасне проведення інструктажів з безпеки праці;

7.3 Пожежна безпека

Достатньо вивісити на видному місці інструкцію із заходів пожежної безпеки (пункт 3.5 Правил пожежної безпеки в Україні). Вогнегасники слід встановлювати в легкодоступних і помітних місцях (у коридорах, біля входів або виходів з приміщень тощо), а також у пожежонебезпечних місцях, де найбільш вірогідна поява осередків пожежі. При цьому необхідно забезпечити їх захист від попадання прямих сонячних променів та безпосередньої (без загороджувальних щитків) дії опалювальних та нагрівальних приладів. Вибір типу та необхідна кількість вогнегасників визначається відповідно до Типових норм належності вогнегасників, затверджених наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 2 квітня 2004 р. № 151. Згідно з пунктом 3.8 Типових норм, громадські та адміністративно-побутові будинки на кожному поверсі повинні мати не менше двох переносних (порошкових, водопінних або водяних) вогнегасників з масою заряду вогнегасної речовини 5 кг і більше. Крім того, слід передбачати по одному вуглекислотному вогнегаснику з величиною заряду вогнегасної речовини 3 кг і більше:

- на 20 м^2 площі підлоги в офісних приміщеннях з ПОЕМ, коморах, електрощитових, вентиляційних камерах і технічних приміщеннях;
- на 50 м^2 площі підлоги приміщень архівів, бібліотек, музеїв.

Приміщення, в яких розміщені ПЕОМ, слід оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-1,4 чи ВВК-2 або один ВВПА-400 на три ПЕОМ, але не менше ніж один вогнегасник зазначених типів на приміщення (п. 3.10 Типових норм). Ця вимога стосується також будівель, споруд та приміщень, обладнаних будь-якими типами установок пожежогасіння, пожежної сигналізації або внутрішніми пожежними кранами (п. 6.4.8 Правил пожежної безпеки в Україні).

План евакуації представлено на рисунку 7.2.

Рисунок 7.2 – План евакуації

ВИСНОВКИ

В ході виконання даного дипломної роботи було розроблено веб-додаток банку. При розробці програмного продукту були враховані плюси і мінуси аналогів. Розроблено діаграми для проектування системи і БД. Написано технічне завдання. Було проведено тестування, яке не виявило недоліків, що свідчить про готовність системи до впровадження.

Реалізовано і успішно протестовані наступні функції: реєстрація користувача, авторизація користувача, особистий профіль користувача, додавання карт, переказ коштів, видалення карти користувачем, перегляд усієї історії по карті, поповнення карти, зняття готівки, додавання новин у стрічку адміністратором. Веб-додаток має велику кількість різних програмних можливостей і орієнтований на користувача, який не має великого досвіду роботи з програмними продуктами такого типу. Веб-додаток безпечний для використання і відкритий для оновлення і доповнень до функціоналу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інструкція red bean php. Режим доступу: https://obninsksite.ru/blog/php-scripts/lesson-redbeanphp
2. Актуальність банківської системи. Режим доступу: sites.google.com/site/bankofyourdreams/actually
3. Актуальність сайтів банку. Режим доступу: beautofan.ru/luchshij-internet-sajt-banka-kachestvo-kontenta-i/
4. «Скриптова мова PHP, теорія і практика». Режим доступу: https://sites.google.com/site/da21svietlova/skriptova-mova-php
5. «СУБД MySQL, бази даних: основні поняття». Режим доступу: http://www.webmasterwiki.ru/mysql.

ДОДАТОК А

ФІЗИЧНЕ ПРОЕКТУВАННЯ БД

```
CREATE TABLE `cards` (
  `id` int(11) UNSIGNED NOT NULL,
  `card_type` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `number` bigint(16) DEFAULT NULL,
  `percent` decimal(10,2) DEFAULT NULL,
  `user_login` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `money` double DEFAULT NULL,
  `credit_limit` int(11) UNSIGNED DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `history` (
  `id` int(11) NOT NULL,
  `card_num` bigint(16) NOT NULL,
  `purpose` varchar(100) NOT NULL,
  `sum` int(11) NOT NULL,
  `date` datetime NOT NULL,
  `type` varchar(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `news` (
  `id` int(11) NOT NULL,
  `text` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `users` (
  `id` int(11) UNSIGNED NOT NULL,
  `email` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `password` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `fam` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `name` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
`otchestvo` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

ДОДАТОК Б

```
<?php
$data = $_POST;
$errors = array();
if( isset($data['do_login']) )
{
$user = R::findOne('users', 'email = ?', array($data['email']) );
if ($data['email'] == "admin@mail.ru" && $data['password'] == "admin") { ?>
<script type="text/javascript">
window.location = "admin.php"
</script>
<?php }
else if( $user )
{
if( password_verify($data['password'], $user->password) )
{
$_SESSION['logged_user'] = $user;
echo '<div> Çäðàâñòâóéòå ' .$_SESSION['logged_user']->email. '</div>'; ?>
<script type="text/javascript">
window.location = "index.php"
</script>
<?php
} else
{
$errors[] = 'Íåâåðíûé ïàðîëü';
}
} else
{
$errors[] = 'Ïîëüçîâàòåëü ñ òàêèì E-mail íå íàéäåí';
}
if( !empty($errors) )
{
echo '<div id="errors">' .array_shift($errors).  '</div><hr>';
}}
?>
```

```
<?php
$data = $_POST;
$errors = array();
if( isset($data['do_signup']) )
{
if( trim($data['fam']) == '' )
{
$errors[] = 'Ââåäèòå ôàìèëèþ';
}
if( trim($data['name']) == '' )
{
$errors[] = 'Ââåäèòå èìÿ';
}
if( trim($data['otchestvo']) == '' )
{
$errors[] = 'Ââåäèòå îò÷åñòâî';
}
if( $data['email'] == '' )
{
$errors[] = 'Ââåäèòå Email';
}
if( $data['password'] == '' )
{
$errors[] = 'Ââåäèòå ïàðîëü';
}
if( $data['password2'] != $data['password'] )
{
$errors[] = 'Ïîâòîðíûé ïàðîëü ââåäåí íå âåðíî';
}
if( R::count('users', "email = ?", array($data['email'])) > 0  )
{
$errors[] = 'Ïîëüçîâàòåëü ñ òàêèì Email óæå ñóùåñòâóåò';
}
if( empty($errors) )
{
```

```
$user = R::dispense(users);
$user->email = $data['email'];
$user->password = password_hash($data['password'], PASSWORD_DEFAULT);
$user->fam = $data['fam'];
$user->name = $data['name'];
$user->otchestvo = $data['otchestvo'];
R::store($user);
?>
<script type="text/javascript">
window.location = "index.php"
</script>
<?php
} else {
echo '<div class="center red" id="errors"><strong>' .array_shift($errors).
'</strong></div><hr>';
}}
?>

<?php
if( isset($data['add_card_confirm']) )
{
$max_card = R::findLast('cards');
$foo = bcadd($max_card->number, 1);
if( $_SESSION['c_name'] == 'credit')
{
unset ($_SESSION['c_name']);
$card = R::dispense(cards);
$card->user_login = $_SESSION['logged_user']->email;
$card->pin = password_hash($data['pin'], PASSWORD_DEFAULT);
$card->card_type = 'credit';
$card->number = $foo;
$card->percent = 0.24;
$card->money = 1000;
$card->credit_limit = 1000;
R::store($card);
```

```
        echo '<hr class="hrr" ><div class="center green">Êàðòà ñ íîìåðîì ' .$foo. ' ñîçäàíà</div>';
        } else if( $_SESSION['c_name'] == 'puyer' ) {
        unset ($_SESSION['c_name']);
        $card = R::dispense(cards);
        $card->user_login = $_SESSION['logged_user']->email;
        $card->pin = password_hash($data['pin'], PASSWORD_DEFAULT);
        $card->card_type = 'puyer';
        $card->number = $foo;
        $card->percent = 0.5;
        $card->money = 0;
        $card->credit_limit = 0;
        R::store($card);
        echo '<hr class="hrr" ><div class="center green">Êàðòà ñ íîìåðîì ' .$foo. ' ñîçäàíà</div>';
        } else if ( $_SESSION['c_name'] == 'nakopit' ) {
        unset ($_SESSION['c_name']);
        $card = R::dispense(cards);
        $card->user_login = $_SESSION['logged_user']->email;
        $card->pin = password_hash($data['pin'], PASSWORD_DEFAULT);
        $card->card_type = 'nakopit';
        $card->number = $foo;
        $card->percent = 0.5;
        $card->money = 0;
        $card->credit_limit = 0;
        R::store($card);
        echo '<hr class="hrr" ><div class="center green">Êàðòà ñ íîìåðîì ' .$foo. ' ñîçäàíà</div>';
        }
        }
?>

        <?php
        mysql_connect($host, $username, $pass);
        mysql_select_db($dbname);
        $data = $_SESSION['logged_user']->email;
        $res = mysql_query('select `number`, `card_type`, `money` from `cards` where
        user_login="'.$data.'"');
```

```
while($row = mysql_fetch_assoc($res)){
if($row['card_type'] == 'credit') {
   ?>
   <option class="red" value="<?php echo $row['number'];?>"><?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
   <?php } else if($row['card_type'] == 'nakopit') { ?>
   <option class="green" value="<?php echo $row['number']; ?>"> <?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
   <?php } else if($row['card_type'] == 'puyer') { ?>
   <option class="yellow" value="<?php echo $row['number'];?>"><?php echo
$row['number'];?> <?php echo $row['money']; ?>ãðí</option>
   <?php }
}
?>

<?php $data = $_POST;
   ?>

<select name="transfer_c1" class="center">
<option value="<?php echo $data['transfer_c1']; ?>"><?php echo $data['transfer_c1'];
?></option>
<?php
mysql_connect($host, $username, $pass);
mysql_select_db($dbname);
$data_mail = $_SESSION['logged_user']->email;
$res = mysql_query('select `number`, `card_type`, `money` from `cards` where
user_login="'.$data_mail.'"');
while($row = mysql_fetch_assoc($res)){
if($row['card_type'] == 'credit') {
   ?>
   <option class="red" value="<?php echo $row['number']; ?>"><?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
   <?php  } else if($row['card_type'] == 'nakopit') {
   ?>
```

```
    <option class="green" value="<?php echo $row['number']; ?>"><?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
    <?php } else if($row['card_type'] == 'puyer') {
    ?>
    <option class="yellow" value="<?php echo $row['number']; ?>"><?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
    <?php }
}
?>

<?php
$summa = $data['sum'];
if( isset($data['do_transfer3'] )) {
$bank_card = R::findOne('cards', 'number=?', array('1000000000000000'));
$card_transfer1 = $_SESSION['card_transfer1'];
$card_transfer2 = $_SESSION['card_transfer2'];
$percent = $_SESSION['percent'];
$today = date("Y-m-d H:i:s");
// äîáàâëåíèå èñòîðèè
//êàðòà îòïðàâèòåëÿ
$story = R::dispense(history);
$story->card_num = $card_transfer1->number;
$story->purpose = "Ïåðåâîä íà êàðòó " . $card_transfer2->number;
$story->sum = $data['sum'];
$story->date = $today;
$story->type = '0';
R::store($story);
$story = R::dispense(history);
$story->card_num = $card_transfer2->number;
$story->purpose = "Ïåðåâîä ñ êàðòû " . $card_transfer1->number;
$story->sum = $data['sum'];
$story->date = $today;
$story->type = '1';
R::store($story);
// ---
```

```
unset ($_SESSION['card_transfer1']);
unset ($_SESSION['card_transfer2']);
unset ($_SESSION['percent']);
$card_transfer1->money = round($card_transfer1->money - $data['sum'] - $percent, 2);
$card_transfer2->money = round($card_transfer2->money + $data['sum'], 2);
$bank_card->money = $bank_card->money + $percent;
R::store($bank_card);
R::store($card_transfer1);
R::store($card_transfer2);
echo "<hr><p class='green'>Äåíüãè ïåðåâåäåíû</p>"; }3
if( isset($data['do_transfer']) )
{
if( $data['transfer_c2'] != $data['transfer_c1']) {
if( $data['transfer_c1'] > 0) {
if( $data['transfer_c2'] > 0) {
$card_transfer1 = R::findOne('cards', 'number=?', array($data['transfer_c1']));
$card_transfer2 = R::findOne('cards', 'number=?', array($data['transfer_c2']));
//ïîäñ÷åò ïðîöåíòà
if($card_transfer1->card_type == 'credit') {
$percent = $data['sum'] / 100 * 4;
} else if ($card_transfer1->card_type == 'puyer') {
$percent = $data['sum'] / 100 * 0.5;
} else if ($card_transfer1->card_type == 'nakopit') {
$percent = $data['sum'] / 100 * 0.5;
}
//---
if( $card_transfer1->money-$percent >= $data['sum']) {
if( $data['sum'] > 0 ) {


$_SESSION['card_transfer1'] = $card_transfer1;
$_SESSION['card_transfer2'] = $card_transfer2;
$_SESSION['percent'] = $percent;
```

```
echo "<p>Êîìèññèÿ: ".$percent."</p><label class='center label button_add_card'><input
type='submit' name='do_transfer3' value=">Ïîäòâåðäèòü</label> ";
} else { echo "<hr><p class='red'>Íå âûáðàíà ñóììà</p>";}
} else { echo "<hr><p class='red'>Íåäîñòàòî÷íî ñðåäñòâ</p>";}
} else { echo "<hr><p class='red'>Íå âûáðàíà êàðòà äëÿ ïåðåâîäà</p>";}
} else { echo "<hr><p class='red'>Íå âûáðàíà êàðòà</p>";}
} else { echo "<hr><p class='red'>Êàðòà äëÿ ïåðåâîäà äîëæíà áûòü äðóãàÿ</p>"; } }?>

<?php $data = $_POST; ?>

<select name="transfer_c3" class="center">
<option value="<?php echo $data['transfer_c3']; ?>"><?php echo $data['transfer_c3'];
?></option>
<?php
mysql_connect($host, $username, $pass);
mysql_select_db($dbname);
$data_mail = $_SESSION['logged_user']->email;
$res = mysql_query('select `number`, `card_type`, `money` from `cards` where
user_login="'.$data_mail.'"');
while($row = mysql_fetch_assoc($res)){
if($row['card_type'] == 'credit') {
   ?>
   <option class="red" value="<?php echo $row['number']; ?>"><?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
   <?php } else if($row['card_type'] == 'nakopit') {
   ?>
   <option class="green" value="<?php echo $row['number'];?>"><?php echo
$row['number'];?> <?php echo $row['money']; ?>ãðí</option>
   <?php } else if($row['card_type'] == 'puyer') {
   ?>
   <option class="yellow" value="<?php echo $row['number']; ?>"><?php echo
$row['number']; ?> <?php echo $row['money']; ?>ãðí</option>
   <?php }
}
?>
```

```
<?php
$summa = $data['sum2'];
$bank_card = R::findOne('cards', 'number=?', array('1000000000000000'));

if( isset($data['do_transfer_num_confirm']) ) {
$card_transfer3 = $_SESSION['card_transfer3'];
$card_transfer4 = $_SESSION['card_transfer4'];
$percent = $_SESSION['percent'];
$today = date("Y-m-d H:i:s");
// äîáàâëåíèå èñòîðèè
//êàðòà îòïðàâèòåëÿ
$story = R::dispense(history);
$story->card_num = $card_transfer3->number;

if( $data['nazn'] != '') {
$story->purpose = 'Ïåðåâîä ïî íîìåðó êàðòû ' . $card_transfer4->number . ' Ñîîáùåíèå: ' .
$data['nazn'];
} else {$story->purpose = 'Ïåðåâîä ïî íîìåðó êàðòû ' . $card_transfer4->number;}

$story->sum = $data['sum2'];
$story->date = $today;
$story->type = '0';
R::store($story);
//êàðòà ïîëó÷àòåëÿ
$story = R::dispense(history);
$story->card_num = $card_transfer4->number;

if( $data['nazn'] != '') {
$story->purpose = 'Ïåðåâîä ïî íîìåðó êàðòû îò ' . $card_transfer3->number . ' Ñîîáùåíèå: ' .
$data['nazn'];
} else {$story->purpose = 'Ïåðåâîä ïî íîìåðó êàðòû îò ' . $card_transfer3->number;}

$story->sum = $data['sum2'];
$story->date = $today;
```

```
$story->type = '1';
R::store($story);
// ---
unset ($_SESSION['card_transfer3']);
unset ($_SESSION['card_transfer4']);
unset ($_SESSION['percent']);
$card_transfer3->money = round($card_transfer3->money - $data['sum2'] - $percent, 2);
$card_transfer4->money = round($card_transfer4->money + $data['sum2'], 2);
$bank_card->money = $bank_card->money + $percent;
R::store($bank_card);
R::store($card_transfer3);
R::store($card_transfer4);
echo "<hr><p class='green'>Äåíüãè ïåðåâåäåíû</p>"; }

if( isset($data['do_transfer2']) )
{
if( $data['transfer_c3'] != $data['num_c']) {
if( $data['transfer_c3'] > 0) {
if( $data['num_c'] > 0) {
$card_transfer3 = R::findOne('cards', 'number=?', array($data['transfer_c3']));
$card_transfer4 = 0;
$card_transfer4 = R::findOne('cards', 'number=?', array($data['num_c']));
if( $card_transfer4 != 0) {
//ïîiäñ÷åò ïðîöåíòà
if($card_transfer3->card_type == 'credit') {
$percent = $data['sum2'] / 100 * 4;
} else if ($card_transfer3->card_type == 'puyer') {
$percent = $data['sum2'] / 100 * 0.5;
} else if ($card_transfer3->card_type == 'nakopit') {
$percent = $data['sum2'] / 100 * 0.5;
}

//---
if( $card_transfer3->money-$percent >= $data['sum2']) {
if( $data['sum2'] > 0 ) {
```

```
$_SESSION['card_transfer3'] = $card_transfer3;
$_SESSION['card_transfer4'] = $card_transfer4;
$_SESSION['percent'] = $percent;
$pers = R::findOne('users', 'email=?', array($card_transfer3->user_login));
echo "<p>Êàðòà: ".$data['num_c']."</p> <p>".$pers->fam." ".$pers->name." ".$pers->otchestvo."</p>
<input class='nazn center' type='text' name='nazn' placeholder='Íàçíà÷åíèå ïëàòåæà'>
<p>Êîìèññèÿ: ".$percent." </p> <label class='center label button_add_card'><input type='submit' name='do_transfer_num_confirm' value=''>Ïîäòâåðäèòü</label> ";

} else { echo "<hr><p class='red'>Íå âûáðàíà ñóììà</p>";}
} else { echo "<hr><p class='red'>Íåâîñòàòî÷íî ñðåäñòâ</p>";}
} else { echo "<hr><p class='red'>Êàðòû ñ íîìåðîì ".$data['num_c']." íå ñóùåñòâóåò</p>";}
} else { echo "<hr><p class='red'>Íå âûáðàíà êàðòà äëÿ ïåðåâîäà</p>";}
} else { echo "<hr><p class='red'>Íå âûáðàíà êàðòà</p>";}
} else { echo "<hr><p class='red'>Êàðòà äëÿ ïåðåâîäà äîëæíà áûòü äóãàÿ</p>"; } }
?>
```