**Итоговый отчет**

***Team name:***

ibanking-pro

***Team:***

Белоус Станислав, [programer911@gmail.com](http://vk.com/write?email=programer911@gmail.com)  
Козловский Эдвардс, edward2769940@gmail.com

Македонов Денис, [denis.makedonov@gmail.com](http://vk.com/write?email=denis.makedonov@gmail.com)  
Плетнев Роман. [pletnevroman95@gmail.com](http://vk.com/write?email=pletnevroman95@gmail.com)

**User stories:**

|  |  |  |  |
| --- | --- | --- | --- |
| User | Story | Operation | Developer |
| Оператор | Войти в систему | Включить приложение и ввести свой логин и пароль | Плентнев Роман |
| Оператор | Зачислить деньги на определенный аккаунт определенного пользователя | Ввести команду: пополнение баланса. Ввести логин пользователя и идентификационный номер аккаунта, а так же сумму | Македонов Денис |
| Оператор | Снять деньги с конкретного аккаунта определенного пользователя | Ввести команду в консоль: снятие денег. Ввести логин пользователя, идентификационный номер аккаунта, а так же сумму. | Македонов Денис |
| Администратор | Войти в систему | Включить приложение и ввести свой логин и пароль | Плентнев Роман |
| Администратор | Добавить Пользователя | Ввести команду: добавление пользователя, ввести новый логин и пароль. | Белоус Станислав |
| Администратор | Удалить пользователя | Ввести команду: удаление пользователя и ввести логин | Белоус Станислав |
| Администратор | Добавить аккаунт | Ввести команду добавление аккаунта, вести тип и дополниетльные данные. | Белоус Станислав |
| Администратор | Удалить аккаунт | Ввести команду удаление аккаунта у пользователя, ввести номер аккаунта и логин пользователя | Белоус Станислав |
| Администратор | Добавление денег на аккаунт | Ввести команду добавление денег, ввести номер аккаунта и сумму | Македонов Денис |
| Администратор | Снятие денег со счета | Ввести команду снятия денег со счета, ввести номер аккаунта и сумму. | Македонов Денис |
| Администратор | Перевод денег | Ввести команду перевода денег, ввести номер аккаунта откуда, куда и сумму. | Македонов Денис |

Git repository:

<https://github.com/tpuser/ibanking-pro>

TargetProcess:

<https://ibanking.tpondemand.com>

Account - password:

Stas - qwerty123

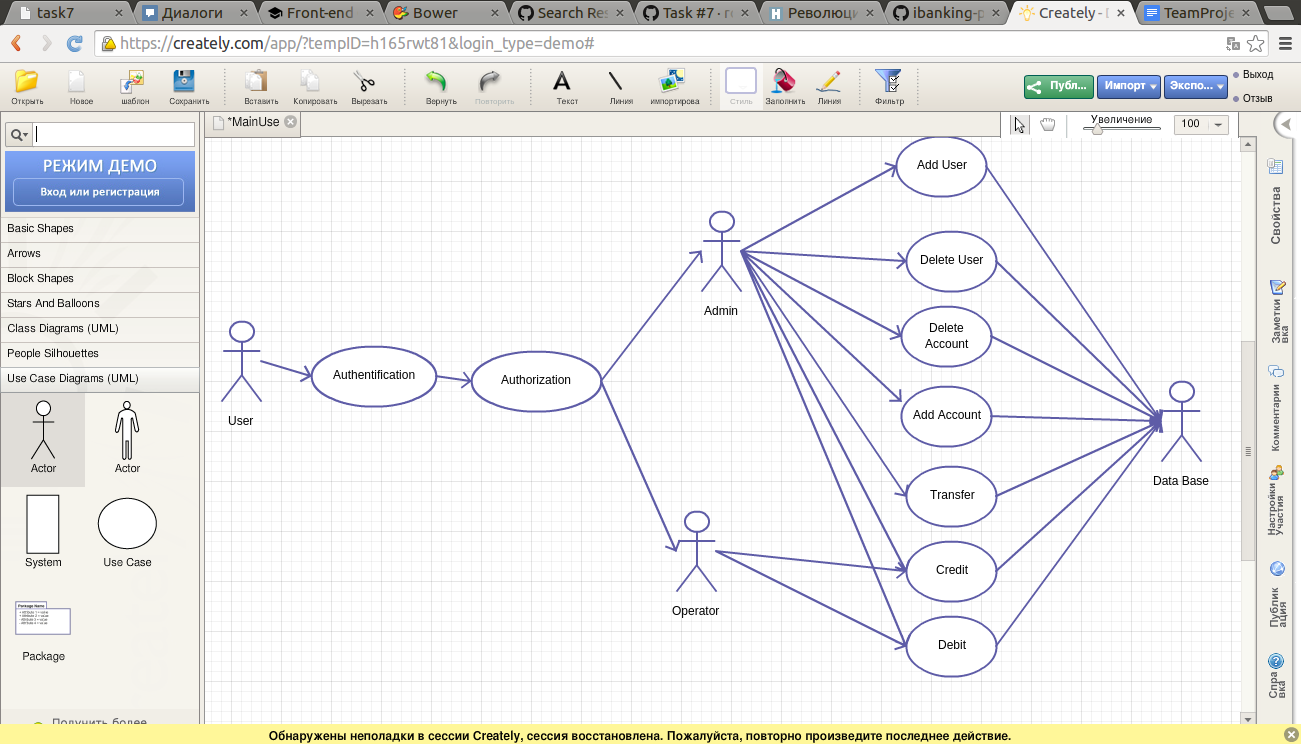
Edwards - qwerty123

Denis - qwerty123

Roman - qwerty123

Архитектура приложения:

Данное приложение реализует следующую диаграмму вариантов использования:



***Общая схема приложения:***

Любой пользователь, запустивший приложение должен пройти аутентификацию и авторизацию. После удачного прохождения данных процедур, пользователь получает доступ к командной консоли. Далее пользователь имеет возможность ввести команду, которую он хочет выполнить. После нажатия клавиши ‘enter’ происходит проверка наличия данной команды в системе, и в случае, если она есть, то начнется ее выполнение.

Перечисление всех команд находится в файле ‘commands.h’, а исполняемый код в ‘commands.c’. Так же в приложении объявления всех функции, предоставляющие интерфейс доступа к базу данных расположены в файле ’dbinterface.h’, а их реализация - ‘dbinterface.c’.

***Как собрать и запустить проект:***

1. склонить проект с гита:

git clone <https://github.com/tpuser/ibanking-pro.git>

1. перейти в папку с проектом:

cd ibanking-pro/admin-console

1. собрать проект:

make

1. запустить:

./admin-console

1. (опционально) запустить тесты:

./test

1. ***Белоус Станислав*** :

**Имя функции** *deleteAccount*

Служит для удаления аккаунта из базы данных

Как работает

На вход фунции поступает база данных из которой мы хотим удалить аккаунт. Приложение запрашивает с консоли Id аккаунта после чего выполняется запрос на удаление аккаунта по данному Id.

Реализация

printf("Enter ACCOUNT ID >> ");

scanf("%d", &idAccount);

sprintf(sql, "DELETE FROM account WHERE (accountID = '%d');", idAccount);

rc = sqlite3\_exec(db, sql, 0, 0, &zErrMsg);

**Имя функции** *addAccount*

Служит для добавления нового аккаунта в базу данных

Как работает

С консоли запрашивается логин пользователя к которому мы хотим добавить аккаунт. Выполняется запрос на существование такого пользователя, если такой не существует, то в консоль выводится сообщение и прекращается работа функции, иначе консоль потребует выбрать тип аккаунта, который мы хотим добавить, после чего, в зависимости от типа аккаунта консоль запросит информацию для аккаунта, после этого будет выполнена команда для создания нового аккаунта.

Реализация

printf("Enter LOGIN >> ");

scanf("%s", newLogin);

checkLogin = -1;

sprintf(sql, "SELECT customerID FROM customers WHERE (login = '%s');", newLogin);

rc = sqlite3\_exec(db, sql, callbackLoginCustomer, (void\*)&checkLogin, &zErrMsg);

if(checkLogin == -1){

printf("Not login in database\n");

}

else {

for(;;) {

printf("\nChoose type account\n0- Checking\n1 - Saving\n2 - OverDraft\n");

scanf("%d", &typeAccount);

if(typeAccount >= 0 && typeAccount <=2)

break;

}

printf("Enter BALANCE new account >> ");

scanf("%f", &balance);

if(balance < 0)

balance = 0;

if(typeAccount == 0)

sprintf(sql, "INSERT INTO account(balance, totalTransaction, accountTypeID, customerID) VALUES('%f', '0', '%d', '%d');", balance, typeAccount, checkLogin);

else if(typeAccount == 1)

sprintf(sql, "INSERT INTO account(balance, accountTypeID, customerID) VALUES('%f', '%d', '%d');", balance, typeAccount, checkLogin);

else if(typeAccount == 2)

sprintf(sql, "INSERT INTO account(balance, time, accountTypeID, customerID) VALUES('%f', '0', '%d', '%d');", balance, typeAccount, checkLogin);

rc = sqlite3\_exec(db, sql, 0, 0, &zErrMsg);

}

**Имя функции** createNewCustomer

Служит для добавления нового пользователя в базу данных

Как работает

Консоль запрашивает новый логин и пароль нового пользователя после чего создается и выполняется запрос к базе данных, который добавляет нового пользователя в таблицу пользователей базы данных.

Реализация

printf("Enter LOGIN and PASSWORD across space>> ");

scanf("%s %s", newLogin, newPassword);

sprintf(sql, "INSERT INTO customers(login, password) VALUES('%s', '%s');", newLogin, newPassword);

rc = sqlite3\_exec(db, sql, 0, 0, &zErrMsg);

**Имя функции** deleteCustomer

Служит для удаления пользователя из базы данных

Как работает

Консоль запрашивает логин пользователя, которого мы хотим удалить. Выполняется запрос на получение Id данного пользователя. Удаляем данного пользователя и все его аккаунты из базы данных.

Реализация

printf("Enter LOGIN for DELETE >> ");

scanf("%s", newLogin);

sprintf(sql, "SELECT customerID FROM customers WHERE (login = '%s');", newLogin);

rc = sqlite3\_exec(db, sql, callbackLoginCustomer, (void\*)&checkLogin, &zErrMsg);

//Delete customer's accounts

sprintf(sql, "DELETE FROM account WHERE (customerID = '%d');", checkLogin);

rc = sqlite3\_exec(db, sql, 0, 0, &zErrMsg);

//Delete custоmer

sprintf(sql, "DELETE FROM customers WHERE (login = '%s');", newLogin);

rc = sqlite3\_exec(db, sql, 0, 0, &zErrMsg);

Выполнил:

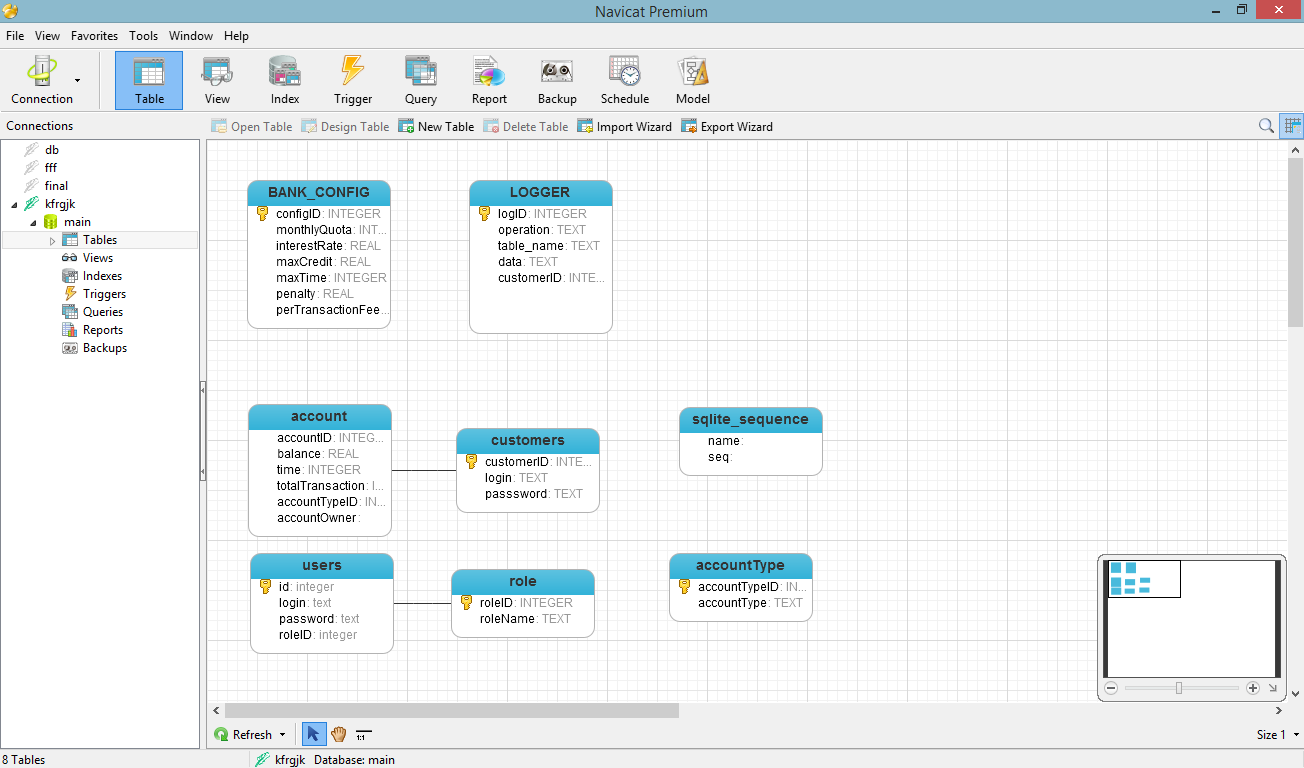
* функцию оздания аккаунта
* функцию удаления аккаунта
* функцию удаления пользователя со всеми его счетами
* функцию добавления нового пользователя в базу данных

1. ***Козловский Эдвардс***

Выполнил:

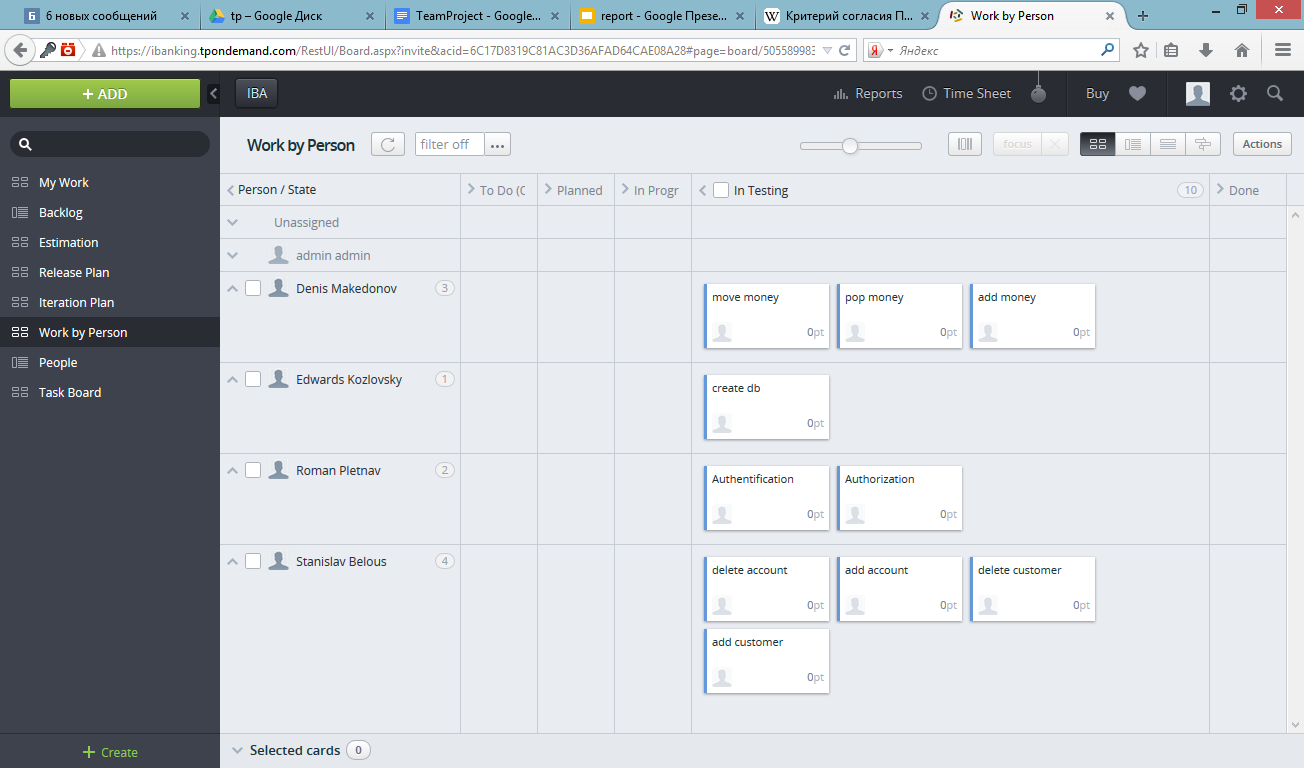
* составление списка user stories
* создание базы данных

схема:



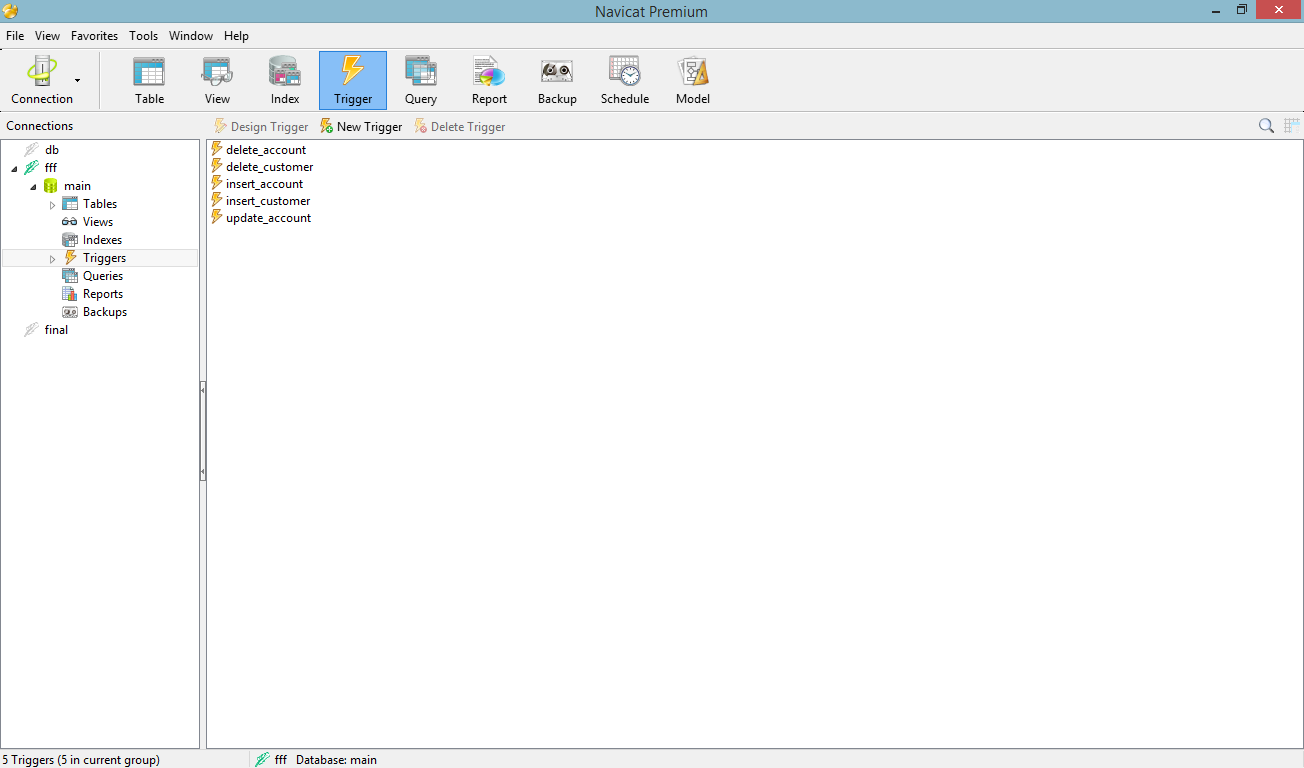
* мониторинг проекта на TargetProcess:

1. создание проекта.
2. регистрация пользователей
3. мониторинг изменений.



* создание журнала транзакций:

1. Добавление триггеров в базу данных



1. Вывод информации из соответствующей таблицы данных

|  |
| --- |
| bool showLogger(sqlite3 \* db, char \* result){ |
| bool finalResult = false; |
| sqlite3\_exec(db, SHOW\_LOGGER, callbackShowInfo, (void\*)result, NULL); |
|  |
| return finalResult; |
| } |

1. ***Плетнев Роман***

Выполнил:

* создание и поддержка проекта на github

ссылка на проект - <https://github.com/tpuser/ibanking-pro>

* реализация аутентификации/авторизации в системе

кусочек хитрого кода, который скрывает ввод пароля:

void setStdinEcho(bool enable)

{

#ifdef WIN32

HANDLE hStdin = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD mode;

GetConsoleMode(hStdin, &mode);

if(!enable)

mode &= ~ENABLE\_ECHO\_INPUT;

else

mode |= ENABLE\_ECHO\_INPUT;

SetConsoleMode(hStdin, mode );

#else

struct termios tty;

tcgetattr(STDIN\_FILENO, &tty);

if( !enable )

tty.c\_lflag &= ~ECHO;

else

tty.c\_lflag |= ECHO;

(void) tcsetattr(STDIN\_FILENO, TCSANOW, &tty);

#endif

}

printf("login: ");

while (scanf("%"XSTR(MAX\_LOGIN\_SIZE)"s", login) != 1) {}

while (!loginExists(db, login))

{

printf("login %s does not exist\n", login);

printf("login: ");

while (scanf("%"XSTR(MAX\_LOGIN\_SIZE)"s", login) != 1) {}

}

// password

retryCount = 1;

printf("password: ");

setStdinEcho(false); // hide password input

while (scanf("%"XSTR(MAX\_PASSWORD\_SIZE)"s", password) != 1) {}

setStdinEcho(true);

printf("\n");

// ...

* реализация обработки ввода команд пользователя

(тут ничего особо интересного):

while (1)

{

printf("%s> ", login);

if (fgets(command, MAX\_COMMAND\_SIZE, stdin) == NULL)

continue;

// get first word from command string

cmdLength = 0;

while ((command[cmdLength] != '\0') && (!isspace(command[cmdLength])))

cmdLength++;

if (cmdLength == 0) // empty command

continue;

cmd = malloc(cmdLength + 1);

strncpy(cmd, command, cmdLength);

cmd[cmdLength] = '\0';

// check command

if (!commandExists(cmd))

{

printf("%s: command not found\n", cmd);

free(cmd);

continue;

}

// check permissions

if (!checkCommandPermission(db, login, cmd))

{

free(cmd);

printf("sorry, you don't have permission\n");

continue;

}

// check exit command

if (!strcmp(cmd, COMMAND\_EXIT))

{

free(cmd);

sqlite3\_close(db);

return 0;

}

// run command

executeCommand(db, command);

free(cmd);

}

* создание Makefile

########################################################################

# Makefile for building: admin-console

########################################################################

####### Compiler, tools and options

CC = gcc

CFLAGS = -Wall

LIBS = -pthread -ldl

####### Files

SOURCES = main.c \

testing.c \

commands.c \

dbinterface.c \

admin\_command\_cust.c \

sqlite3.c

OBJECTS = commands.o \

dbinterface.o \

admin\_command\_cust.o \

sqlite3.o

TARGET = admin-console

TEST\_TARGET = test

####### Build rules

all: $(TARGET) $(TEST\_TARGET)

clean:

$(RM) -f $(TARGET) $(TEST\_TARGET) \*.o \*~

$(TEST\_TARGET): testing.o

$(CC) $(CFLAGS) testing.o $(OBJECTS) $(LIBS) -o $(TEST\_TARGET)

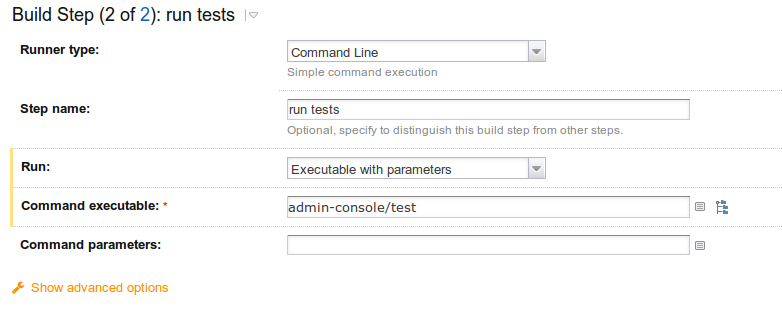
$(TARGET): $(OBJECTS) main.o

$(CC) $(CFLAGS) main.o $(OBJECTS) $(LIBS) -o $(TARGET)

%.o: %.c

$(CC) $(CFLAGS) -c $<

* создание тестов и интеграция тестирования в TeamCity



1. ***Македонов Денис***

Выполнил:

* реализация функций транзакций:

void commit(sqlite3 \*db)

{

managTransaction(db, 0);

}

void undo(sqlite3 \*db)

{

managTransaction(db, 1);

}

void debit(sqlite3 \*db, int acc\_id, double sumtoget)

{

char \* acc, \* sum;

acc = calloc(100, sizeof(char));

sum = calloc(100, sizeof(char));

sprintf(acc, "%i", acc\_id);

sprintf(sum, "%lf", - sumtoget);

updateBalance(db, acc, sum);

free(acc);

free(sum);

}

void credit(sqlite3 \*db, int acc\_id, double sumtopush)

{

char \* acc, \* sum;

acc = calloc(100, sizeof(char));

sum = calloc(100, sizeof(char));

sprintf(acc, "%i", acc\_id);

sprintf(sum, "%lf", sumtopush);

updateBalance(db, acc, sum);

free(acc);

free(sum);

}

void transfer(sqlite3 \*db, int acc\_from, int acc\_to, double sum)

{

debit(db, acc\_from, sum);

credit(db, acc\_to, sum);

//credit(db, acc\_to, sum);

}

void checkAccount (sqlite3 \*db, int acc\_id)

{

double sum;

char \* acc;

acc = calloc(100, sizeof(char));

sprintf(acc, "%i", acc\_id);

checkBalance(db, acc, &sum);

free(acc);

printf("balance (of id = %i) = %lf\n", acc\_id, sum);

}

void showAll(sqlite3 \* db)

{

char \* res;

res = calloc(2000, sizeof(char));

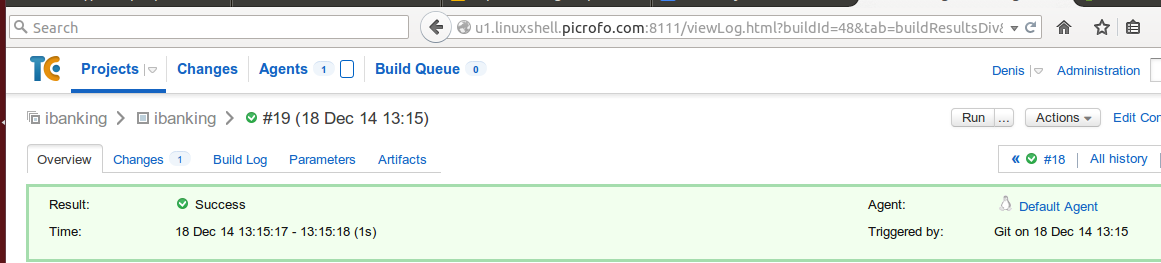
strcat(res, "\n");

showInfo(db, res);

printf("%s", res);

}

* финальное тестирование и отлаживание программы.
* менеджмент
* организация непрерывной интеграции на удаленном сервере



* тестирование проекта
* рефакторинг и поддержка