Пояснительная записка

При запуске(запускается файл menu.py) открывается главное меню(мой класс menuWidget - наследник QWidget), на котором можно создать аккаунт или выбрать одну из игр(НИМ, сапер, крестики-нолики), также там можно увидеть таблицы рейтинга. Действия реализованы при помощи QPushButton, а рейтинги при помощи QPlainTextEdit. Рейтинги представлены в читабельном виде, для которого было выставлено ограничение по длине имени пользователя и пропуски между каждым местом в таблице. Рейтинги обновляются при запуске приложения и после окончания каждой игры путем получения информации из SQL таблицы.

При нажатии кнопки создания аккаунта открывается новое окно, в котором нужно ввести имя аккаунта и пароль. Ограничения на имена: уникальность и длина менее 25 символов. Ограничений на пароль нет, за исключением того, что пароль должен быть непустым. Все проверки реализуются с помощью try, except, raise. Проверка на уникальность реализована с помощью получения имен из SQL таблицы. Об ошибке или о создании пользователю сообщается через специальный QLabel внизу окна. Есть две кнопки “OK” и “Cancel”, первая из которых начинает попытку создания аккаунта, а вторая - закрывает окно без создания. При успешном создании в специальном QLabel отображается “Account created”, в SQL таблицу добавляется аккаунт, и диалог блокируется. Пользователь должен выйти из диалога, чтобы продолжить работу с программой.

При нажатии кнопки какой-либо из игр открываются соответствующие диалоговые окна, реализующие подготовку к игре, а именно ввод данных аккаунтов и, для сапера, задание размера поля. В сапере вводится один аккаунт, а в других играх два. Для конфиденциальности символы в паролях при вводе не отображаются. В сапере задание размера производится при помощи QSpinBox, где минимум 5, а максимум 13(сделано, чтобы поле имело бомбы, то есть игра имела смысл, и не выходило за экран; используется именно QSpinBox, чтобы не давать пользователю возможности ввести неправильные данные). Для проверки на соответствие пароля имени берется информация из SQL таблицы по имени и настоящий пароль сравнивается с введенным паролем. Проверки реализуются при помощи try, except, raise. Как и у окна создания аккаунтов, общение с пользователем происходит через специальный QLabel внизу окна, а также имеются кнопки “OK” и “Cancel” выполняющие функции подтверждения и отмены соответственно.

Так как вышеперечисленные 4 диалоговых окна имеют некоторые одинаковые функции был создан класс myDialog - наследник QDialog, который реализует эти функции, а те 4 окна являются наследниками уже myDialog. Инициализация интерфейса и задание одинаковых связей сигналов и слотов происходит в инициализации myDialog, а специфика инициализируется в потомках. Для инициализации интерфейса в инициализацию myDialog передается имя ui файла, которые нужно загрузить. А также в myDialog имеется функция проверки совпадения пароля с настоящим, в которую надо передать имя и пароль.

При успешном входе в аккаунты в подготовительных окнах запускаются сами игры:

1. НИМ: имеется один класс, реализующий и интерфейс, и саму игру. Чтобы пользователь не мог взять неверное количество камней комбинируются QSpinBox и QRadioButton. При помощи второго пользователь выбирает кучу, из которой хочет взять камни, а при помощи первого выбирает их количество. Каждый раз при выборе кучи обновляется максимум и минимум в QSpinBox, где минимум 1, а максимум - количество камней в куче. Если в куче становится 0 камней, то она блокируется и выбирается другая куча автоматически. Проигрывает тот, кто не может сделать ход. После окончание игры окно блокируется, в SQL таблице изменяется рейтинг(+1 победителю), внизу окна выводится имя победителя, чтобы продолжить взаимодействие пользователю нужно закрыть окно. Также внизу таблицы можно увидеть имя ходящего игрока во время игры.
2. Крестики-нолики: имеется один класс, реализующий и интерфейс, и игру. После выбора клетки, она блокируется. Вверху окна можно увидеть текущий знак. Внизу окна можно увидеть имя ходящего игрока. После каждого хода выполняется проверка на выигрыш. После окончания игры внизу окна можно увидеть имя победителя, в SQL таблице изменяется рейтинг(+1 победителю), и окно блокируется, для продолжения взаимодействия см. окончание НИМа.
3. Сапер: интерфейс реализуется двумя моими классами CoordButton(наследник QPushButton, хранящий в себе координаты кнопки) и QtSapper(сам интерфейс, видный пользователю). Ходы происходят при помощи нажатия на закрытую кнопку поля. Вверху окна можно выбрать тип хода при помощи QRadioButton. При ходе генерируется запрос для функции do\_query, которая осуществляет взаимодействие между интерфейсом и классами, реализующими игру. В ней происходит маркирование или удаление маркировки, а также открытие нажатой клетки. При открытии происходит проверка на выигрыш и проигрыш, а также само открытие. Открытая клетка блокируется. Реализуют механику игры два класса Board(который является доской) и Cage(каждая клетка поля), с которыми и взаимодействует наша функция. После взаимодействия с классами, работающими с механикой, функция обновляет интерфейс. При победе/поражении функция изменяет рейтинг игрока в SQL таблице(+кол-во клеток поля/-кол-во открытых клеток поля). После окончания игры окно блокируется, внизу него можно увидеть сообщение о победе/проигрыше. Для продолжения взаимодействия см. окончание крестиков-ноликов.

В каждую из игр приходят данные об имени игроков, для отображения ходов и изменения рейтинга. В каждый из классов для интерфейса приходит курсор SQL для работы с SQL таблицей. Интерфейс некоторых QT классов был сделан в Qt Designer, других - написан вручную. Весь интерфейс сделан с использованием layout-ов, поэтому размер окна можно изменять для своего удобства, интерфейс от этого не пострадает.