МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Дополнительное образование «Основы промышленного программирования»

**Проект PyQt по теме**

***«******Графический метод решения задач линейного программирования в экономических задачах»***

Ученик: Иванкин Е. Г.

Учитель: Копытина Е. А.

Воронеж, 2020

Оглавление

[Название проекта 2](#_Toc55136857)

[Авторы проекта 2](#_Toc55136858)

[Описание идеи 2](#_Toc55136859)

[Описание реализации 2](#_Toc55136860)

[Описание технологий 2](#_Toc55136861)

[Интерфейс программы 2](#_Toc55136862)

# Название проекта

Требуется выполнить проект по QT под названием «Графический метод решения задач линейного программирования в экономических задачах», который обладает следующими функциональными возможностями:

1. Решение задачи линейного программирования с двумя положительными переменными (оптимизационная экономическая задача), просмотр и сохранение отчёта о решении.
2. Добавление, изменение, удаление, просмотр и экспорт объектов, содержащих данные задач.
3. Поиск задач по номеру, условию, тегам.

# Авторы проекта

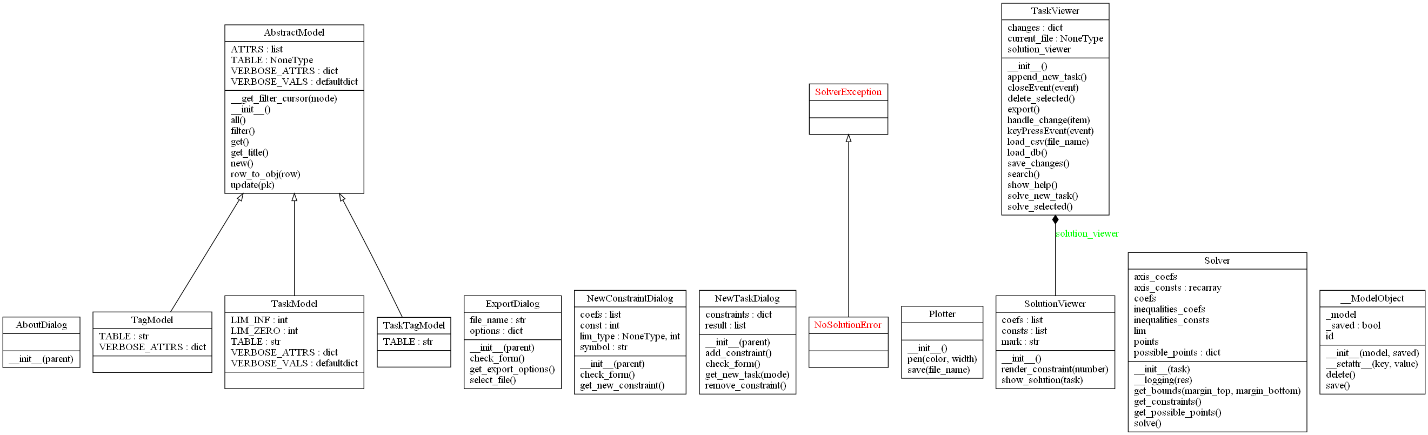
Автором проекта является Иванкин Евгений Геворкович, обучающийся второго года обучения проекта Яндекс.Лицей, который самостоятельно выполнил вышеописанный проект.

# Описание идеи

Графический метод довольно прост и нагляден для решения задач линейного программирования с двумя переменными. Он основан на геометрическом представлении допустимых решений и ЦФ задачи. Для удобства решения и оформления задач линейного программирования графическим методом мною было создано вышеописанное приложение.

# Описание реализации

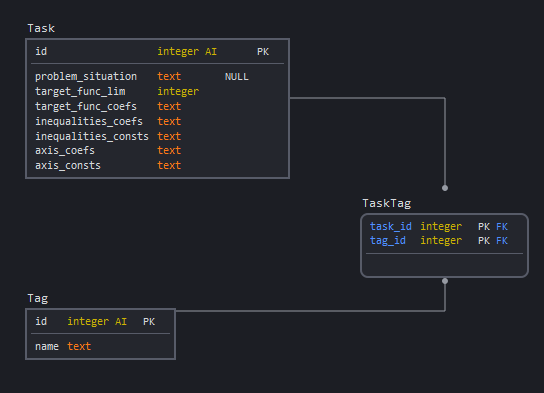
## Основные классы



*Рис. 1 – диаграмма классов*

1. **AboutDialog**: диалоговое окно, отображающее информацию о программе и список горячих клавиш.
2. **ExportDialog**: диалоговое окно для выбора и валидации опций экспорта задач в CSV.
3. **NewConstraintDialog**: диалоговое окно для ввода, валидации и преобразования данных линейного ограничения. Запускается из диалога **NewTaskDialog** при добавлении неравенства.
4. **NewTaskDialog**: диалоговое окно для ввода, валидации и преобразования данных задачи линейного программирования.
5. **SolutionViewer**: окно для взаимодействия с решением задачи. Содержит в себе виджет **Plotter**.
6. **Plotter**: виджет, унаследованный от класса **pyqtgraph.PlotWidget**, сохраняющий в себе некоторые настройки и упрощающий взаимодействие с виджетом-родителем. Позволяет просматривать и экспортировать графическое решение задачи линейного программирования.
7. **TaskViewer**: базовое окно, открывающееся на старте программы. Позволяет загружать задачи из базы данных, CSV, запускать их решение, редактировать и удалять их. Запускает все остальные диалоги и окна.
8. **Solver**: класс для решения задачи линейного программирования и получения данных для его отрисовки. Поднимает исключение **NoSolutionError**, если не удаётся найти решение задачи. При возникновении других проблем при решении (когда решатель из библиотеки scipy возвращает ненулевой код) поднимает исключение **SolverException**. Используется в **SolutionViewer**.
9. **NoSolutionError**: исключение, унаследованное от **SolverException** (которое, в свою очередь, наследуется от встроенного класса **Exception**).

## База данных и связанные классы



*Рис. 2 – структура базы данных*

1. **AbstractModel**: базовый класс модели (часть простой ORM) для наследования от неё реальных моделей, соответствующих сущностям в базе данных. Позволяет добавлять объекты, получать, удалять их и изменять, получать и изменять значения атрибутов через соответствующие методы. Содержит в себе внутренний приватный класс **ModelObject** для разделения методов, относящихся к объекту и всей таблице БД.
2. **TaskModel**: модель, унаследованная от **AbstractModel**, соответствующая таблице Task в базе данных. Объект модели – задача линейного программирования
3. **TagModel**: модель, унаследованная от **AbstractModel**, соответствующая таблице Task в базе данных. Объект модели – тег (служит для удобства и разграничения поиска по задачам).
4. **TaskTagModel**: модель, унаследованная от **AbstractModel** и связывающая теги с задачами (реализация связи многие-ко-многим).

# Описание технологий

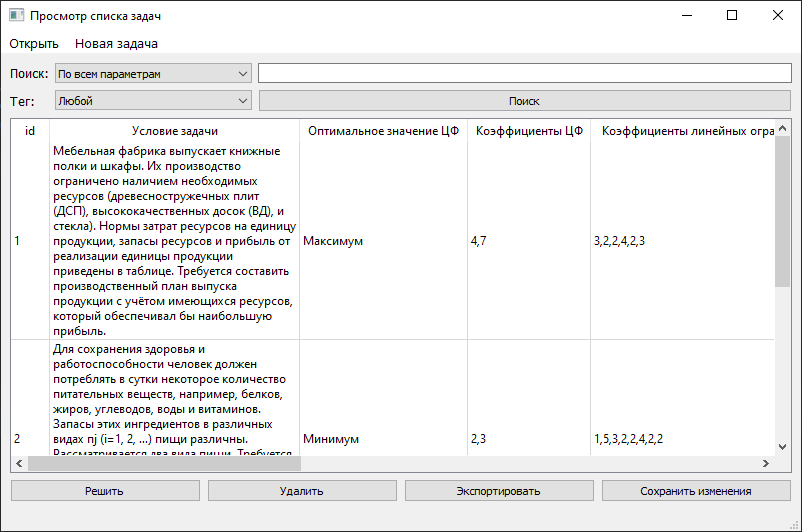
В разработанном приложении «Графический метод решения задач линейного программирования в экономических задачах» реализованы все рассмотренные в ходе изучения блока QT технологии:

1. Программное создание интерфейса (заполнение и очистка таблицы).
2. Разработка интерфейса средствами QT Designer (окно просмотра списка задач, формы для экспорта и создания задач).
3. Использование исключений (подъём и обработка встроенных исключений, использование своих исключений, унаследованных от встроенных).
4. Работа с диалоговыми окнами и изображениями (выбор файлов).
5. Обработка нажатий клавиш (см. список горячих клавиш в разделе «О программе», который можно вызвать клавишей F1)
6. Работа с файлами (чтение и запись в CSV)
7. Использование БД (см. структуру выше).
8. Для визуализации решений использована библиотека PyQtGraph.

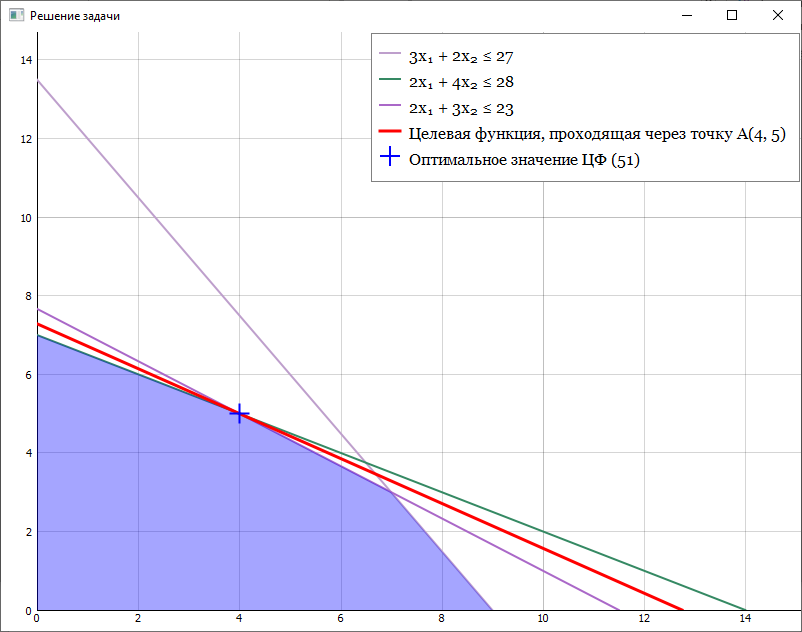
Для запуска требуются следующие библиотеки:

1. csv
2. PyQt5
3. pyqt5-sip >= 12.8, < 13
4. pyqtgraph
5. scipy
6. numpy >= 1.14.5

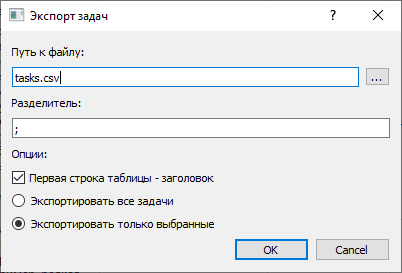
# Интерфейс программы



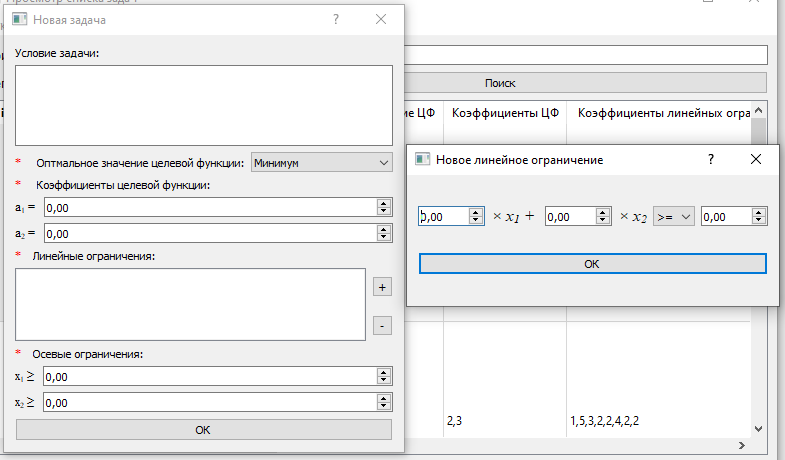
*Рис. 3 – окно просмотра списка задач*



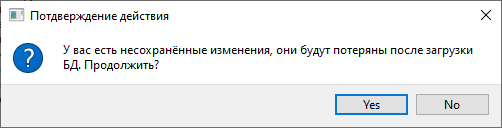
*Рис. 4 – окно просмотра решения задачи*



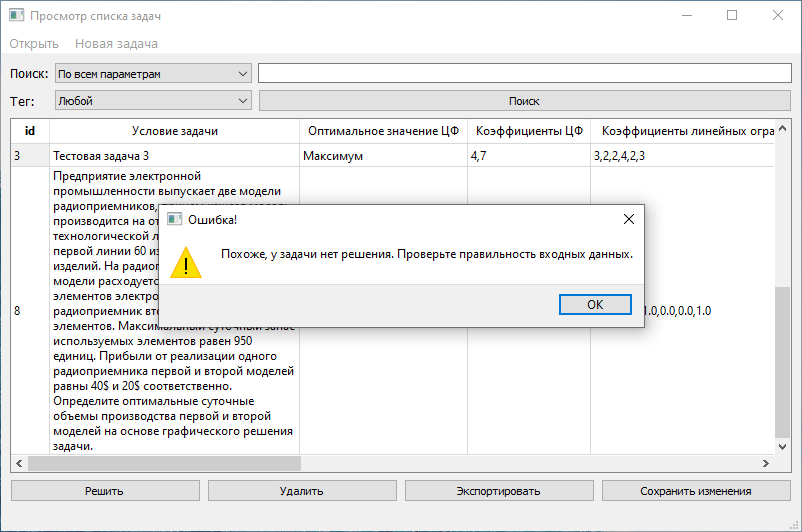
*Рис. 5 – диалог экспорта задач в CSV*



*Рис. 6 – диалоговые окна для заполнения данных задачи*



*Рис. 7 – один из типов уведомлений*



*Рис. 8 – обработка случая, когда у задачи нет решения*