Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ — Информатика и системы управления
КАФЕДРА ИУ7 — Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент

Маслова Марина Дмитриевна

Группа

ИУ7-83Б

Тип практики

преддипломная практика

Название предприятия

НУК ИУ МГТУ им. Н. Э. Баумана

Студент	_____	**Маслова М. Д.**
Руководитель практики от предприятия	_____	**Оленев А. А.**
Руководитель практики	_____	**Кострицкий А. С.**

Оценка_____

2023 г.

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
_____ Рудаков И. В.
«13» мая 2023 г.

З А Д А Н И Е
на прохождение производственной практики (преддипломная практика)

Студент **4** курса группы **ИУ7-83Б**

Маслова Марина Дмитриевна

в период с 15.05.2023 г. по 28.05.2023 г.

Предприятие: **НУК ИУ МГТУ им. Н. Э. Баумана**

Руководитель практики от предприятия (наставник):

Оленев А. А.

Руководитель практики от кафедры:

Кострицкий А. С.

Задание:

1. Изучить программные средства проектирования и разработки информационных систем.

2. Собрать материалы в области разработки информационных систем.

3. Получить практические навыки в области разработки информационных систем.

Дата выдачи задания «13» мая 2023 г.

Руководитель практики от кафедры _____ **Кострицкий А. С.**

Руководитель практики от предприятия _____ **Оленев А. А.**

Студент _____ **Маслова М. Д.**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во время выполнения выпускной квалификационной работы был разработан метод построения поисковых индексов в реляционной базе данных на основе глубоких нейронных сетей.

Целью данной работы является разработка программного обеспечения, демонстрирующего практическую осуществимость спроектированного в ходе выполнения выпускной квалификационной работы метода.

Для достижения поставленной цели требуется решить следующие задачи:

— описать формальную постановку задачи построения индекса в реляционной базе данных на основе глубоких нейронных сетей;
— описать ограничения, накладываемые на входные и выходные данные;
— отразить основные этапы метода в виде IDEF0-диаграмм;
— описать структуру программного обеспечения;
— описать выбор средств программной реализации;
— разработать программное обеспечение, реализующее метод построения индекса.

1 Основная часть

1.1 Формальная постановка задачи

Для построения поискового индекса в реляционной базе данных на основе нейронной сети требуется:

— исходный набор ключей в качестве входных данных;
— набор указателей на данные, соответствующие ключам;
— правила их предварительной обработки, требующейся для обучения модели;
— модель нейронной сети в качестве основы будущего индекса;
— алгоритм обучения нейронной сети, результатом работы которого является обученная модуль, представляющая собой индекс.

Формально данная задача может быть описана с помощью IDEF0-диаграммы нулевого уровня, представленной на рисунке 1.1.

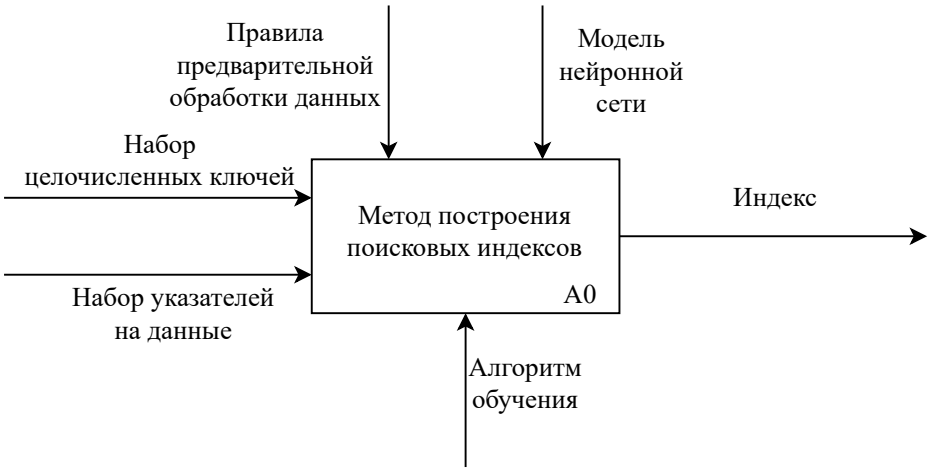

Рисунок 1.1 – Постановка задачи

1.2 Ограничения на входные и выходные данные

На входные данные метода накладываются следующие ограничения:

— в качестве ключей на вход принимаются целые числа для исключения решения дополнительной задачи преобразования входных данных;
— ключи во входном наборе уникальны.

Результатом работы метода является структура данных, представляющая собой индекс над таблицей реляционной базы данных, построенные на основе

глубокой нейронной сети. То есть выходным данным является структура, имеющая следующие поля:

— отсортированный массив ключей, поданных на вход;
— отсортированный по значениям ключей массив указателей на данные, соответствующие ключам;
— модель обученной глубокой нейронной сети, с помощью которой будет предсказываться положение ключа в отсортированном массиве;
— средняя и максимальная абсолютные ошибки предсказания позиции ключа, для ее уточнения и возврата верного указателя на данные.

Краткое описания индекса, являющегося результатом работы метода, как структуры данных представлено на рисунке 1.2.

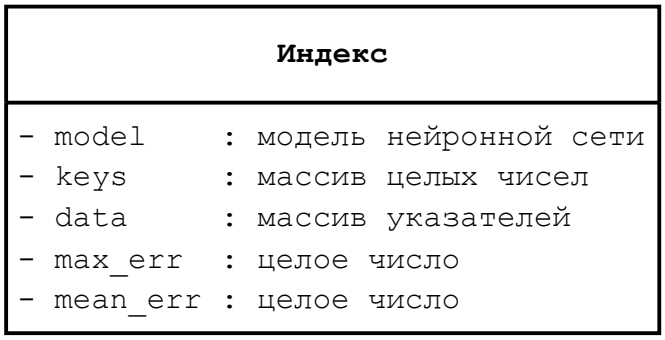

Рисунок 1.2 – Индекс как структура данных

Так как индекс строится над таблицей реляционной базы данных накладываются ограничения на виды поддерживаемых SQL-запросов. Разрабатываемое программное обеспечение должно предоставлять возможность работы с запросами фильтрации, представленными оператором `WHERE` следующими с условием:

— `column operator value`,
где `column` — имя проиндексированного столбца,
`operator` — одна из операций сравнения: `=, <, >, <=, >=`,
`value` — некоторое целочисленное значение.
— `column BETWEEN value1 AND value2`,
где `column` — имя проиндексированного столбца,
`value1, value2` — целочисленные значения, представляющие нижнюю и верхнюю границы диапазона.

1.3 Основные этапы метода

Основные этапы метода построения индекса приведены на IDEF0-диаграмме первого уровня (рисунок 1.3).

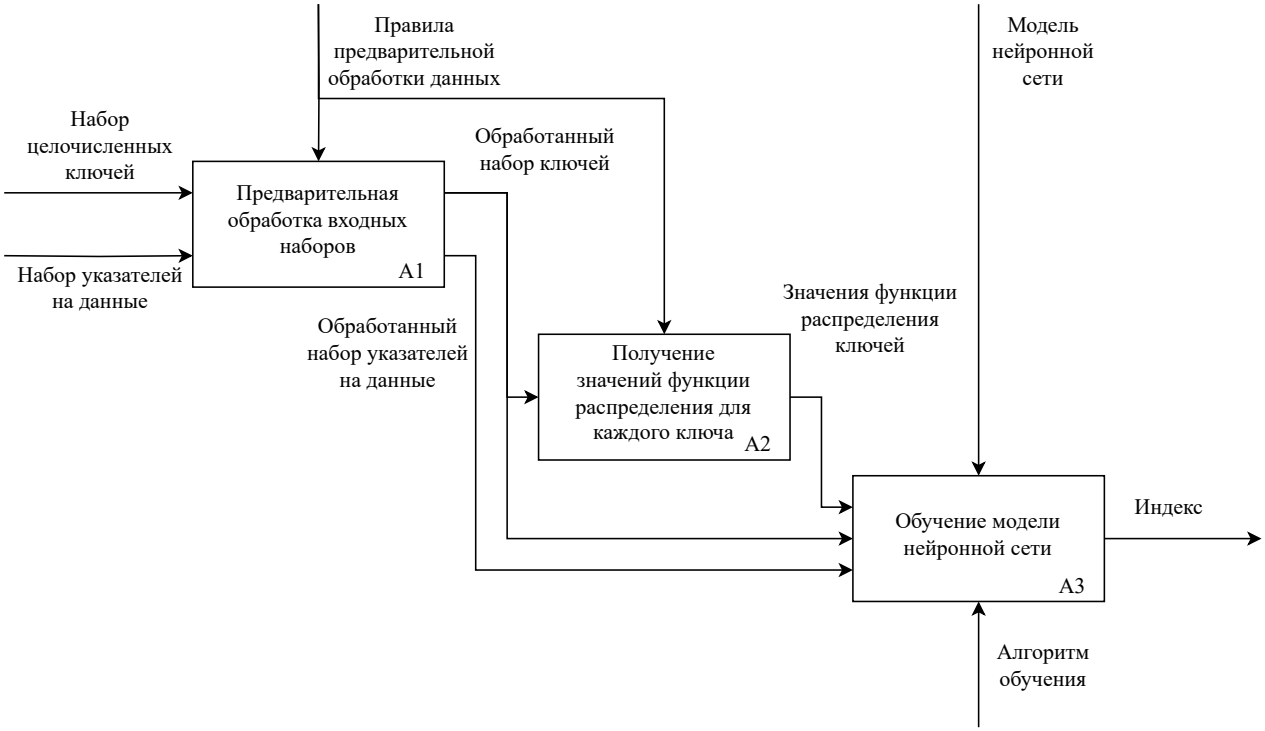

Рисунок 1.3 – IDEF0-диаграмма метода построения индекса в реляционной базе данных на основе глубоких нейронных сетей

Для проверки работоспособности метода также разработка основной операции, осуществляемой с помощью индекса — поиска, IDEF0-диаграмма алгоритма которого представлена на рисунках 1.4-1.5.

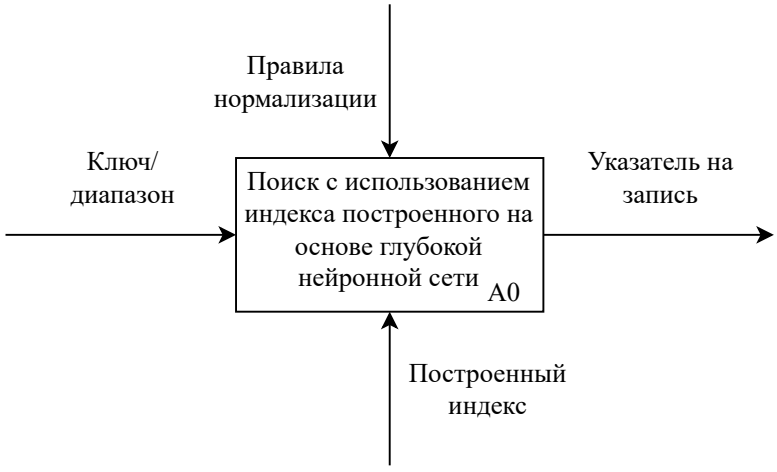

Рисунок 1.4 – IDEF0-диаграмма нулевого уровня поиска

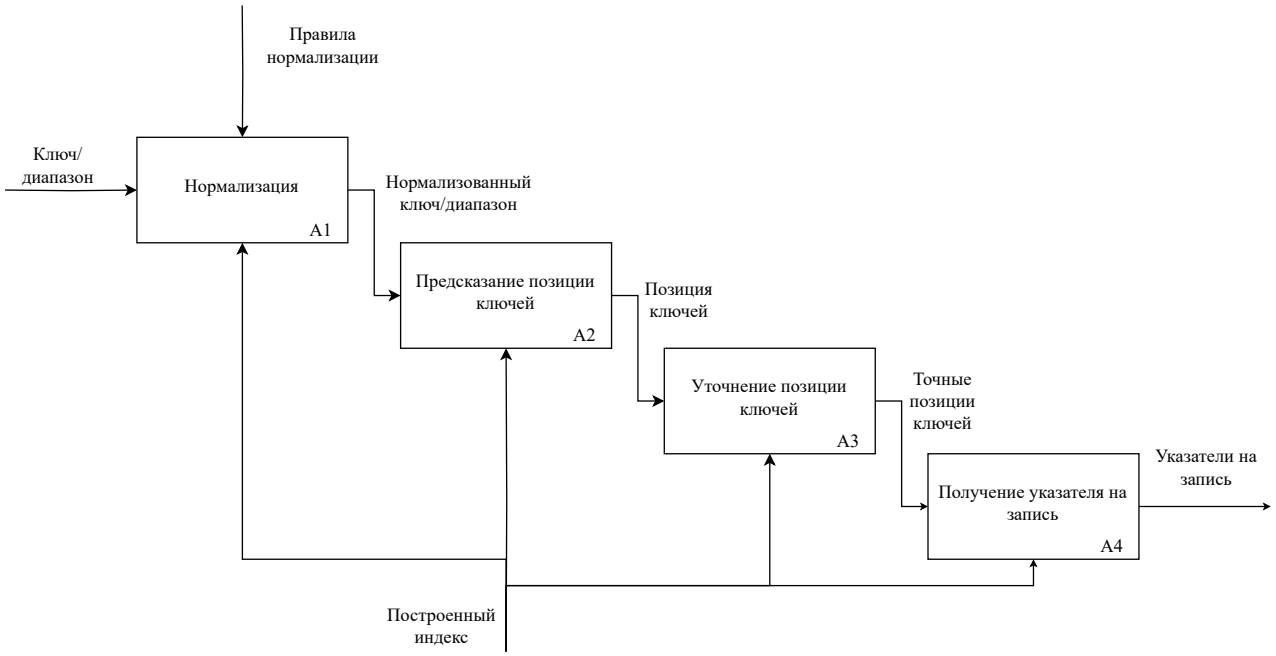

Рисунок 1.5 – IDEF0-диаграмма первого уровня поиска

1.4 Структура программного обеспечения

Программное обеспечение включает в себя три модуля: для взаимодействия с пользователем, взаимодействия с реляционной базой данных и собственно модуля, реализующего индекс. Подробная схема взаимодействия модулей представлена на рисунке 1.6.

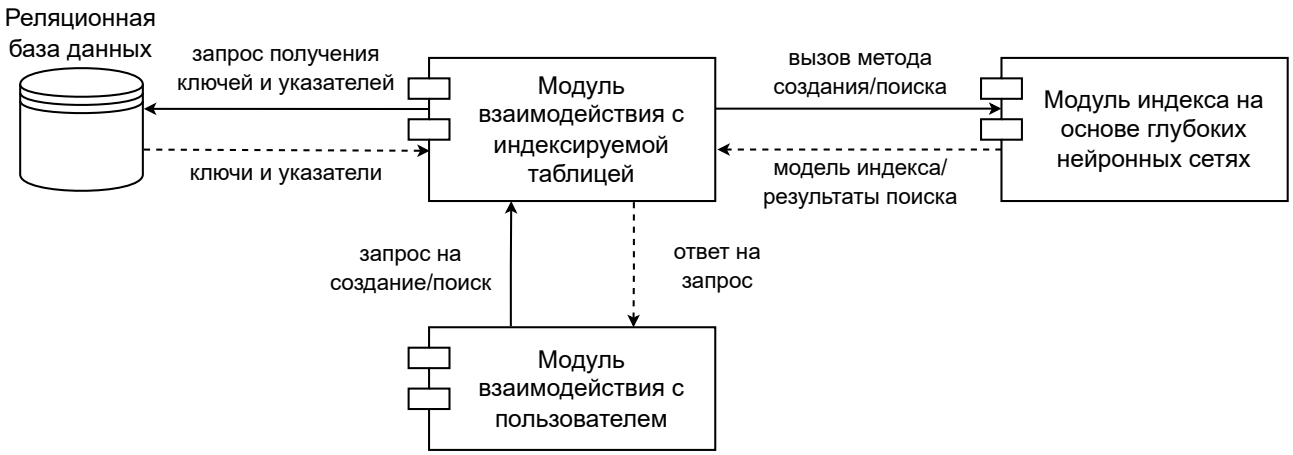

Рисунок 1.6 – Структура программного обеспечения

1.5 Выбор средств программной реализации

Для реализации метода построения индекса в реляционной базе данных на основе глубоких нейронных сетей в качестве языка программирования

выбран Python 3.10 [1]. В качестве библиотеки глубокого обучения выбран TensorFlow 2.11.0 [2] и работающий поверх нее высокоуровневый программный интерфейс Keras 2.11.0 [3]. Для работы с массивами данных выбрана библиотека numpy [4].

В качестве реляционной системы управления базами данных выбрана SQLite [5], предоставляющая программный интерфейс виртуальных таблиц, позволяющих реализовать пользовательский поисковый индекс. Виртуальные таблицы являются одним из видов расширений SQLite, программный интерфейс которых предоставляется на языке C [6], который и выбран в качестве языка программирования для взаимодействия с реляционной базой данных.

Для обеспечения взаимодействия между компонентом работы с базой данных и компонентом, непосредственно реализующий индекс, используются библиотеки языка C `Python.h` для работы с объектами языка Python и `numpy/arrayobject.h`, предоставляющая программный интерфейс для работы с numpy-массивами, которые являются основным типом данных, через который происходит взаимодействие модулей.

1.6 Ключевые моменты реализации

Основными моментами реализации индекса на основе глубокой нейронной сети являются обучение и поиск, которые представлены на листингах 1.1-1.3. Там же отражены основные поля индекса.

Ключевыми моментами реализации модуля, обеспечивающего взаимодействие индекса и реляционной базы данных, являются функции поиска путем прохода по массиву указателей, полученные с помощью индекса, которые представляют собой методы (листинг 1.4), реализующие виртуальную таблицу в SQLite, а также получения данных из таблицы с преобразованием их в тип данных для индекса (листинг 1.5).

1.7 Взаимодействие с программным обеспечением

Взаимодействие с программным обеспечением происходит через командную строку sqlite3, которая является модулем взаимодействия с пользователем. Пример работы представлен на листинге 1.6.

Листинг 1.1 – Реализация построения индекса

```
def _init_for_train(self, keys: list[int], data: list[any]):
    sort_indexes = np.argsort(keys)

    self.N = len(keys)
    self.keys = np.array(keys)[sort_indexes]
    self.norm_keys = self._normalize(self.keys)
    self.data = np.array(data)[sort_indexes]
    self.positions = np.arange(0, self.N) / (self.N - 1)

def _true_train(self):
    self.model.fit(
            self.norm_keys,
            self.positions,
            batch_size=1,
            callbacks=[LossDiffStop(1e-3), self.metrics],
            epochs=30)

def train(self, keys: list[int], data: list[any]):
    self._init_for_train(keys, data)
    self.metrics = MetricCallback(self.norm_keys, self.positions)
    self._true_train()

    self.mean_abs_err = self.metrics.mean_abs_err
    self.max_abs_err = self.metrics.max_abs_err
    self.trained = True
```

Листинг 1.2 – Реализация поиска с помощью построенного индекса

```
def _predict(self, keys):
    if not self.trained:
        return None

    keys = self._normalize(keys)
    pposition = self.model(keys)
    return np.around(pposition * self.N).astype(int).reshape(-1)

def find(self, keys):
    if not self.trained or not keys:
        return None

    keys = np.array(keys)
    positions = self._predict(keys)
    positions = self._clarify(keys, positions)
    return self.data[positions]
```

Листинг 1.3 – Реализация уточнения позиции

```
def _clarify(self, keys, positions):
    def clarify_one(key, position):
        position = max(min(position, self.N - 1), 0)

        low = max(position - self.mean_abs_err, 0)
        high = min(position + self.mean_abs_err, self.N - 1)

        if not (self.keys[low] < key < self.keys[high]):
            low = max(position - self.max_abs_err, 0)
            high = min(position + self.max_abs_err, self.N - 1)

        return binsearch(self.keys, low, high, key)

    vec_clarify = np.vectorize(clarify_one)
    positions = vec_clarify(keys, positions)
    return positions[positions >= 0]
```

Листинг 1.4 – Реализация работы курсора

```
int lindexNext(sqlite3_vtab_cursor *cur) {
    lindex_cursor *pCur = (lindex_cursor*)cur;
    lindex_vtab *lTab = (lindex_vtab*)cur->pVtab;

    sqlite3_reset(lTab->stmt);
    sqlite3_clear_bindings(lTab->stmt);
    int64_t rowid = *(int64_t *)PyArray_ITER_DATA(pCur->iter);
    sqlite3_bind_int64(lTab->stmt, 1, rowid);
    sqlite3_step(lTab->stmt);

    PyArray_ITER_NEXT(pCur->iter);
    return SQLITE_OK;
}

int lindexColumn(sqlite3_vtab_cursor *cur,
                 sqlite3_context *ctx,
                 int i) {
    lindex_vtab *lTab = (lindex_vtab*)cur->pVtab;
    int columnValue = sqlite3_column_int(lTab->stmt, i);
    sqlite3_result_int(ctx, columnValue);
    return SQLITE_OK;
}

int lindexEof(sqlite3_vtab_cursor *cur) {
    lindex_cursor *pCur = (lindex_cursor*)cur;
    return !PyArray_ITER_NOTDONE(pCur->iter);
}
```

Листинг 1.5 – Инициализация индекса

```
int initPythonIndex(sqlite3 *db,
                    const char *const tableName,
                    const char *const modelName,
                    lindex_vtab *vTab) {
    char* query = sqlite3_mprintf("SELECT ROWID, * FROM %s",
       tableName);

    sqlite3_stmt* stmt;
    sqlite3_prepare_v2(db, query, -1, &stmt, NULL);
    sqlite3_free(query);

    PyObject* builderModule = PyImport_ImportModule("indexes.
       builder");
    PyObject* builderClassName = PyObject_GetAttrString(
       builderModule, "LindexBuilder");
    PyObject* pyModelName = PyTuple_Pack(1, PyUnicode_FromString(
       modelName));
    PyObject* builder = PyObject_CallObject(builderClassName,
       pyModelName);
    PyObject* lindex = PyObject_CallMethod(builder, "build", NULL
       );
    PyObject* keys = PyList_New(0);
    PyObject* rows = PyList_New(0);

    int i = 0;
    while (sqlite3_step(stmt) == SQLITE_ROW) {
        int key = sqlite3_column_int(stmt, 1);
        int64_t rowid = sqlite3_column_int64(stmt, 0);

        PyList_Append(keys, PyLong_FromLong(key));
        PyList_Append(rows, PyLong_FromLong(rowid));

        i++;
    }

    if (i) {
        PyObject* train = PyUnicode_FromString("train");
        PyObject* check = PyObject_CallMethodObjArgs(lindex,
           train, keys, rows, NULL);
        Py_DECREF(check);
        Py_DECREF(train);
    }

    vTab->lindex = lindex;

    Py_DECREF(keys);
    /* ... */
    sqlite3_finalize(stmt);

    return SQLITE_OK;
}
```

Листинг 1.6 – Пример работы программного обеспечения

```
$ sqlite3 test.db
sqlite> .load ./lindex
sqlite> create table maps(keys INTEGER);
sqlite> .mode csv
sqlite> .import osm1000.csv maps
sqlite> create virtual table virtmaps using lindex(0, fcnn2);
Epoch 1/30
1000/1000 [====================] - 1s 768us/step - loss: 0.0011
...
sqlite> select * from virtmaps where keys = 232131750;
keys
232131750
```

ЗАКЛЮЧЕНИЕ

В ходе преддипломной практики было разработано программное обеспечение, демонстрирующее практическую осуществимость спроектированного в ходе выполнения выпускной квалификационной работы метода построения поисковых индексов в реляционной базе данных на основе глубоких нейронных сетей.

Были выполнены следующие задачи:

— описана формальная постановка задачи построения индекса в реляционной базе данных на основе глубоких нейронных сетей;
— описаны ограничения, накладываемые на входные и выходные данные;
— отражены основные этапы метода в виде IDEF0-диаграмм;
— описана структуру программного обеспечения;
— описан выбор средств программной реализации;
— разработано программное обеспечение, реализующее метод построения индекса.

Таким образом, все поставленные задачи были выполнены, а цель достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Welcome to Python [Электронный ресурс]. — URL: https://www.python.org. — (Дата обращения: 17.05.2023).
2. TensorFlow [Электронный ресурс]. — URL: https://www.tensorflow.org/. — (Дата обращения: 17.05.2023).
3. Keras: Deep Learning for humans [Электронный ресурс]. — URL: https://keras.io/. — (Дата обращения: 17.05.2023).
4. NumPy [Электронный ресурс]. — URL: https://numpy.org/. — (Дата обращения: 17.05.2023).
5. SQLite Documentation[Электронный ресурс]. — URL: https://www.sqlite.org/docs.html. — (Дата обращения: 17.05.2023).
6. The GNU C Reference Manual [Электронный ресурс]. — Режим доступа URL: https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html. — (Дата обращения: 17.05.2023).