**Обработка Migrations** 4.1.7

**-Применить все новые миграции:**

php artisan migrate

**-Откатить последнюю (одну) миграцию:**

php artisan migrate:rollback --step=1 – количество откатываемых миграций;

**- Статус созданных миграций:**

php artisan migrate:status

**Create migration**

**-Создать миграцию с именем** create\_migTableName\_table **и кодом создающим таблицу** migTableName **с полями id и timestamps:**

php artisan make:migration create\_migTableName\_table - **имя таблицы** должно быть **во**

**множиственном числе**, слова в имени разделятся \_подчеркиванием;

**-Создать миграцию с именем** migName **и кодом создающим** таблицу tableName **с полями id и timestamps:**

php artisan make:migration migName --create=tableName

**-Создать миграцию с именем** migTableName **и кодом редактирующим существующую таблицу:**

php artisan make:migration migName –table=tableName //так-же см. ниже Update migration

**- Создать пустую миграцию с методами Up и Down:**

php artisan make:migration migName

**Create table fields**

**- Имя столбца** должно быть **в единственном числе**;

**- Типы полей смотри**(вторят типам полей БД)**:**

в главе 4.1.3.1 и [https://laravel.com/docs/9.x/migrations#columns](https://laravel.com/docs/9.x/migrations" \l "columns) и [https://laravel.su/docs/8.x/migrations#columns](https://laravel.su/docs/8.x/migrations" \l "columns)

**- Мягкое удаление:** 4.1.3.2 - не стирает данные а помечает как удаленные;

**- Дополнительные параметры полей:** 4.1.3.3

->default(0) - значение по умолчанию;

->default(new Expression('RAND() ')) - использования SQL выражения;

->nullable() или nullable(true) - поле НЕ ОБЯЗАТЕЛЬНОЕ для заполнения;

->nullable(false) - поле ОБЯЗАТЕЛЬНОЕ для заполнения;

->virtualAs('`price` \* `count`') - вычисляемое поле из полей price и count;

**- Создание индексов (базы данных, для более быстрого поиска):** 4.1.3.4

->index() - создание индекса с алгоритмом по умолчанию;

->index('idx\_name', 'hash') - имя индекса, алгоритм;

->primary() - ключевой индекс;

->unique() - уникальный индекс, поле с таким индексом НЕ МОЖЕТ ПОВТОРЯТСЯ в таблице;

**- Поле внешнего ключа:**4.1.3.5

**$table->foreignId('user\_id')->constrained()->onDelete('cascade')**

->foreignId('user\_id') – создаст столбец 'user\_id' хранящий внешний ключ;

->constrained() - создает связь с другой таблицей, методу можно передать имя первичной таблицы в единственном чиcлe, имя ключевого поля первичной таблицы, но если имя поля внешнего ключа 'user\_id' соответствует формату <имя связываемой первичной таблицы в единственном числе>\_id, а ключевое поле связываемой первичной таблицы называется id, параметры в вызове метода constrained () можно не указывать;

->onDelete('cascade') - удалить связанные записи вторичной таблицы при удалении записи первичной таблицы (у нас -cascade, т. е. каскадное удаление);

**$table->foreignId('user')->constrained('userlist', 'num')** - Поле внешнего ключа user свяжет текущую вторичную таблицу с первичной таблицей userlist, имеющей ключевое поле num;

**$table->unsignedВiginteger('rubric\_id')**

**$table->foreign('rubric\_id')->references('id')->on('rubrics')** - другая форма записи, см стр.110

->onUpdate (<обозначение операции>) - операция, выполняемая при изменении значения ключевого поля у записи первичной таблицы.

**- Текстовая кодировка таблицы** (может конфликтовать с кодировкой БД)**:**

$table->charset = 'utf8mb4'

**Update migration**

**- Создать миграцию изменяющую таблицу:**

php artisan make:migration changesDescription\_to|from|in\_tableName

**Update table fields**

Метод **Down** для **rollback** заполняется в ручную !!!

**- Добавить/удалить поле в таблице:**

up() $table->string('add', 10)->nullable();

down() $table->dropColumn(nameField | arreyNamesFields);

**- Правка параметров поля:**

$table->string('add', 40)->nullable(false)**->change()**;

**- Переименование поля:**

$table->renameColumn('oldName', 'newName');

**- Удаление поля внешнего ключа:** 4.1.4.3

**- Переименование/удаление таблиц:** 4.1.4.4

**- Проверка существования таблицы и полей:** 4.1.5

**Dump** 4.1.8

**- Создание дампа:** (По умолчанию в папке database\schema)

php artisan schema:dump

php artisan schema:dump --prune — после формирования дампа удалить все миграции

**- Накатить дамп:**

php artisan migrate - При этом сначала выполняются SQL-команды из дампа, а потом - не примененные ранее миграции.