

Обработка ошибок

Общие требования

Для работы с ошибками требуется использовать вместо стандартного пакета **errors**, пакет **github.com/pkg/errors**, так как он позволяет учитывать стек вызова, при логировании ошибки.

В любых операциях на уровне выше слоя **storage**, нужно использовать внутренний пакет **apierrors**, для формирования ошибок выдаваемых пользователю. Т.е. на слоях **controller** и **usecase**, а так же в пакете **domain** нужно всегда возвращать соответствующую ошибку из пакета **apierrors**. Если из слоя ниже или другой библиотеки была получена ошибка другого типа, её следует обернуть в ошибку из пакета **apierrors**.

Если соответствующей ошибки нет, ее можно добавить, согласовав назначение нового класса ошибок. Не следует делать специализированные классы ошибок, класс должен отражать широкое множество ошибок схожих по области в которой они возникли. Например, **ValidationError** - класс, отражающий любые ошибки валидации, не следует создавать **MyStructFieldValidationError**, дабы не плодить сущности.

Централизованная обработка ошибок

Условно ошибки можно разделить на два вида:

- Обработанная - ошибка была завернута в **apierror**, значит разработчик предусмотрел её обработку и возврат пользователю.
- Внештатная - любая не обработанная разработчиком ошибка.

Сервис обрабатывает ошибки в едином обработчике, следующим образом:

- Если в обработчик пришла ошибка типа **apierror**, она выводится в лог, если обернутая в нее ошибка поддерживает получение стека-вызовов - стек также запишется в лог. Если не поддерживает, в логе будет только сообщение об ошибке. В логе ошибка помечается признаком **severity=handled**.
- Если пришла ошибка веб-сервера **fiber** - она записывается в лог с **severity=fiber**.
- В остальных случаях, обработчик считает, что ошибка была не обработана и она

записывается в лог с **severity=unhandled**.

При любом варианте записи в лог вызывающему отдается ответ с классом ошибки и причиной её возникновения. Ответ выглядит следующим образом:

```
{
  "success": false,
  "message": "Внутренняя ошибка сервиса",
  "error": {
    "class": "internal",
    "type": "runtime.errorString",
    "reason": "runtime error: integer divide by zero"
  }
}
```

Класс ошибки определяет важность для клиентского приложения, сообщение описывает проблему, объект error показывает причину возникновения ошибки.

Соответствие внутренней и внешней классификации ошибок:

	Класс ошибки apierror
handled	Определяется разработчиком
unhandled	internal
fiber	internal