



# C# / .NET

(chatbot: разработка модулей взаимодействия  
с чатом и доработка доменной логики)

Андрей Голяков

# Доработка схемы компонентов

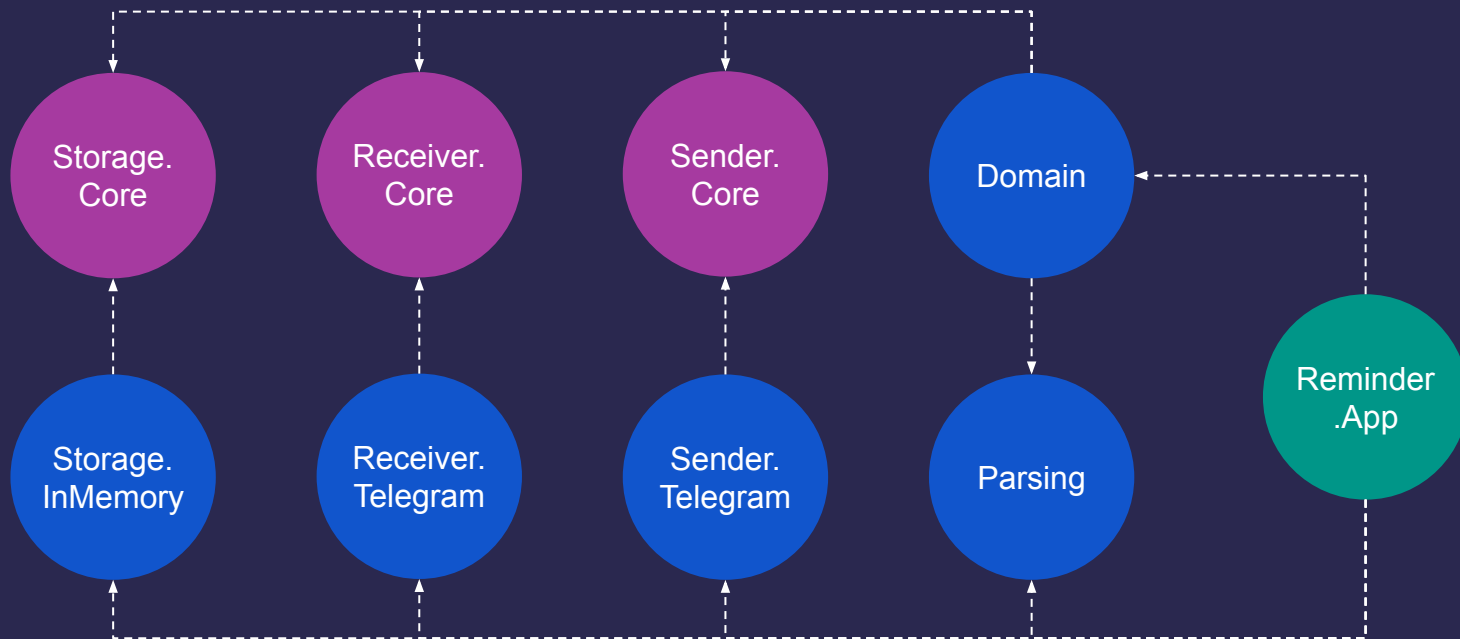
---

Добавляем 5 новых модулей, два отвечающих за взаимодействие с чатом и один за парсинг строк:

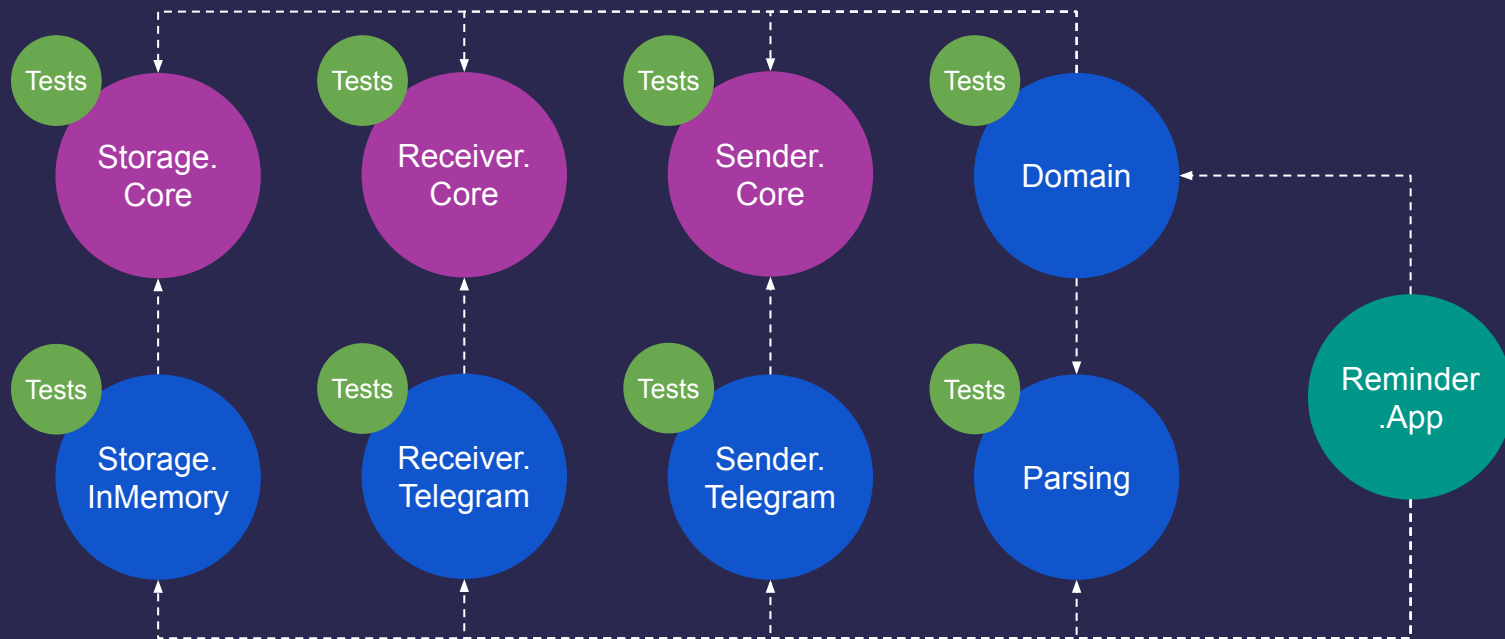
- **Reminder.Receiver.Core**
  - Библиотека с описанием интерфейса классов для получения сообщений с напоминаниями (для последующего добавления их в хранилище).
- **Reminder.Receiver.Telegram**
  - Реализация интерфейса получателя информации о новых напоминаниях от бота Telegram.
- **Reminder.Parsing**
  - Библиотека с реализацией разбора входящих сообщений пользователя.
- **Reminder.Sender.Core**
  - Библиотека с описанием интерфейса классов для отправки напоминаний.
- **Reminder.Sender.Telegram**
  - Реализация интерфейса отправки напоминаний через бота Telegram.



# Взаимосвязи между компонентами



# Мы не забыли про **unit-тесты**\* (см. домашнюю работу)



# Совместная работа в классе

---

Пишем интерфейсы и классы библиотек

- Reminder.**Receiver.Core** (Class Library .NET Standard 2.0.)
- Reminder.**Sender.Core** (Class Library .NET Standard 2.0.)
- Reminder.**Parsing** (Class Library .NET Standard 2.0.)



# Reminder.Receiver.Core

---

- `Storage.Receiver.Core.IReminderReceiver` интерфейс
  - Описывает функционал, отвечающий за приём новых напоминаний.
  - Поскольку приём напоминаний может произойти в любой момент, то в интерфейсе мы опишем
    - Метода “запуска” ожидания новых напоминаний: `Run`
    - Событие “мы получили новое напоминание”: `MessageReceived`
- `Storage.Receiver.Core.MessageReceivedEventArgs` класс
  - Класс аргумента события `MessageReceived`. Содержит следующие члены:
    - Пришедшее текстовое сообщение: `Message`
    - Идентификатор контакта, отправившего сообщение: `ContactId`
    - **Конструктор** с двумя параметрами для инициализации двух свойств выше.



# Reminder.Sender.Core

---

- `Storage.Sender.Core.IReminderSender` интерфейс
  - Описывает функционал, отвечающий за отправку напоминаний.
  - Содержит единственный член: метод `Send`, который принимает два параметра:
    - Идентификатор адресата: `contactId`
    - текстовое сообщение для отправки: `message`



# Reminder.Parsing

---

- `Storage.Parsing.MessageParser` статический класс
  - Отвечает за преобразование текстового сообщения, пришедшего от пользователя в структуру из двух полей:
    - Дата отправки напоминания
    - Собственно, текст напоминания
  - Содержит единственный статический метод: `Parse`
    - возвращает экземпляр класса `ParsedMessage`, описанный ниже.
- `Storage.Parsing.ParsedMessage` класс
  - Класс для возвращения структуры Дата-Сообщение. Содержит эти два свойства:
    - Дата (и время) отправки напоминания: `Date`
    - Текст сообщения: `Message`





# Регистрируем нового бота в Telegram

Находим бота @BotFather, с помощью которого мы будем управлять новым ботом.

- Отправляем команду **/start**
  - Видим много информации о том, что можно делать
- Отправляем команду **/newbot**
  - В ответ получаем “Alright, бла-бла-бла... Введите имя вашего бота.”
- Отправляем текст с именем, **например AGO Reminder Bot**
  - В ответ “Good, теперь введите username вашего бота, он должен заканчиваться на слово ‘bot’. Например TetrisBot или tetris\_bot”
- Отправляем имя пользователя, **например AgoReminderBot**
  - Получаем либо “Sorry...” что означает, что выбранное имя занято, либо “Done! Congratulations... Use this token to access the HTTP API...” и сам токен:  
**000000000:YYYYY\_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**
- Он-то нам и потребуется в коде :)



# Совместная работа в классе

---

Пишем реализацию интерфейсов для получения и отправки напоминаний через Telegram:

- `Reminder.Receiver.Telegram` (Class Library .NET Standard 2.0.)
- `Reminder.Sender.Telegram` (Class Library .NET Standard 2.0.)

В каждый из этих проектов потребуется добавить в зависимости **NuGet-пакет Telegram.Bot** последней версии (при установке со всем соглашаться :)



# Reminder.Receiver.Telegram

## Storage.Receiver.Telegram.TelegramReminderReceiver класс

- Реализует интерфейс `Storage.Receiver.Core.IReminderReceiver`.
- Отвечает за получение сообщений от Telegram-бота.
- В конструкторе принимает токен бота, от которого он будет ждать сообщений и создаёт закрытый член клиента Telegram-бота:

```
public TelegramReminderReceiver(string accessToken)
{
    botClient = new TelegramBotClient(accessToken);
}
```

- В методе `Run` подписываемся на его событие `OnMessage` и запускаем цикл получения:

```
public void Run()
{
    botClient.OnMessage += BotClient_OnMessage;
    botClient.StartReceiving();
}
```

- Далее мы должны пробросить вверх собственное событие `MessageReceived` когда наступит событие клиента `OnMessage`.



# Reminder.Sender.Telegram

## Storage.Sender.Telegram.TelegramReminderSender класс

- Реализует интерфейс `Storage.Sender.Core.IReminderSender`.
- Отвечает за отправку сообщений от имени Telegram-бота.
- В конструкторе принимает токен бота, от имени которого он будет посылать сообщения и создаёт закрытый член клиента Telegram-бота:

```
public TelegramReminderSender(string accessToken)
{
    botClient = new TelegramBotClient(accessToken);
}
```

- В методе Send просто посылаем сообщение:

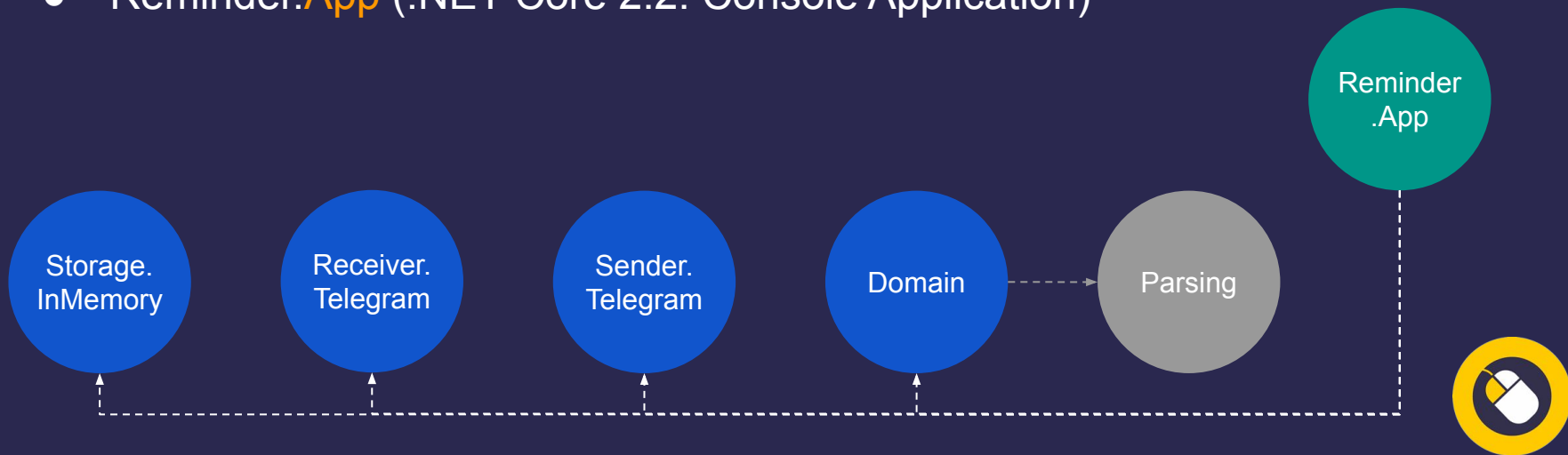
```
public void Send(string contactId, string message)
{
    var chatId = new global::Telegram.Bot.Types.ChatId(long.Parse(contactId));
    botClient.SendTextMessageAsync(
        chatId,
        message);
}
```



# Совместная работа в классе

Модификация доменной логики и написание консольного приложения для работы с Telegram.

- Reminder.**Domain** (Class Library .NET Standard 2.0.)
- Reminder.**App** (.NET Core 2.2. Console Application)



# Домашняя работа

---

Поскольку мы продолжим на протяжении курса работать с этим приложением, рекомендуется сделать следующее:

- Покрыть код юнит-тестами.
- Расширить формат допустимых сообщений для установки новых таймеров.
- Добавить ILogger и его консольную реализацию ConsoleLogger, который бы так же инжектился в конструктор класса доменной логики и выводил в консоль статусные сообщения.

Это позволит в будущем:

- Безболезненно менять одни модули на другие. Например, мы заменим InMemoryReminderStorage на SqlReminderStorage.
- Добавить функционал по сервисному управлению нашим ботом.



# Спасибо за внимание.

