Отправить всё!

Поддержка офлайна в мобильном приложении. Рецепт от Solo

Владимир Абакумов

Цель

Поделиться подходом к неблокирующей отправке созданных пользователем данных на сервер

О чём пойдёт речь

Как «хорошо» отправлять данные на сервер:

- концепция
- варианты реализации
- и ещё пара рецептов

Чего не будет

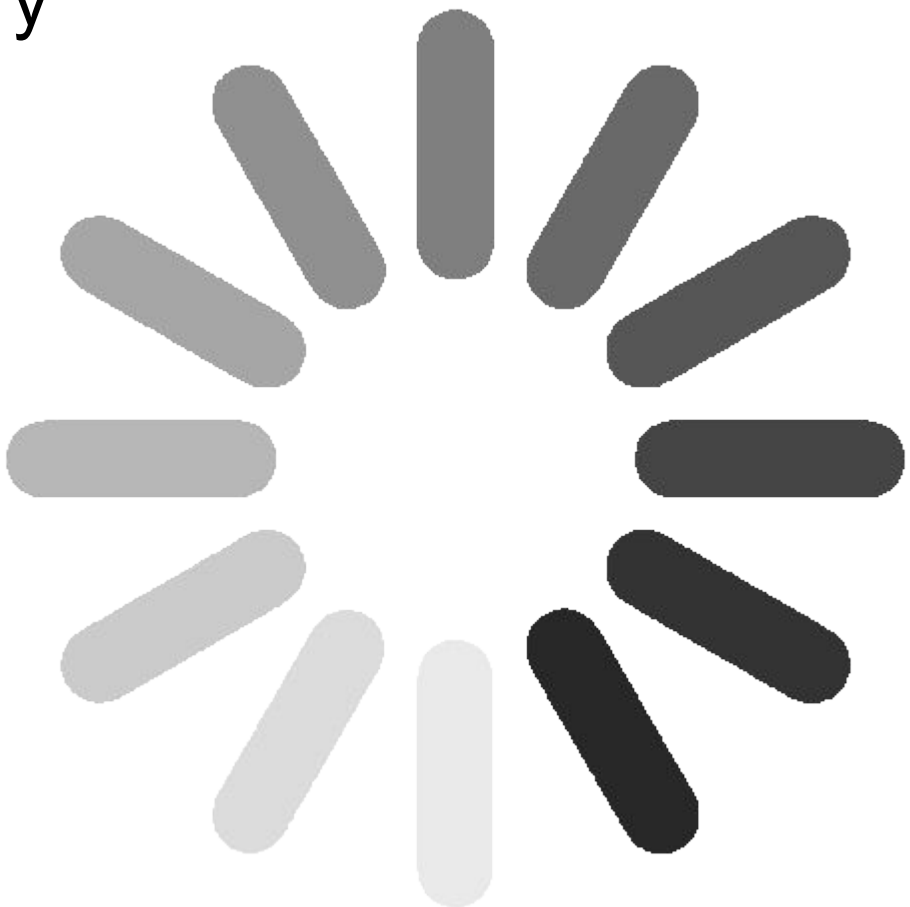
- Транзакционность

Интро

- Владимир Абакумов, TeamLead ОМР
- DIRECTUM и DirectumRX
- DIRECTUM Solo и Jazz

Отправка созданных пользователем данных

- Блокируем работу



Отправка созданных пользователем данных

- Блокируем работу
  - Просто и надежно
  - Ответственность на пользователе

  - Бесяче (ノ ° □ °)ノ (┻━━┻)

Отправка созданных пользователем данных

- Не блокируем работу

Когда узнаешь, что данные пропали

Отправка созданных пользователем данных

- Не блокируем работу
  - Не понятен статус

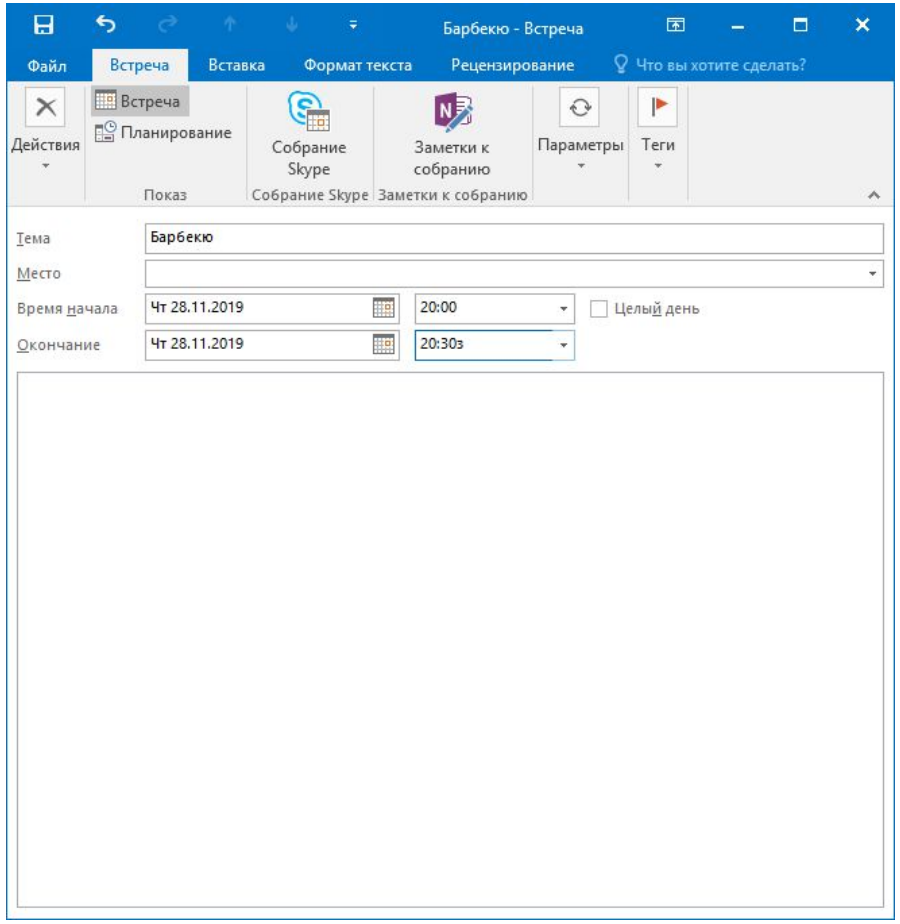
Отправка созданных пользователем данных

- Не блокируем работу
  - Не понятен статус
  - Можно потерять данные

Отправка созданных пользователем данных

- Не блокируем работу
- Не теряем данные

Приложение для организации событий



Приложение для организации событий

- Создаём событие, указывая реквизиты, участников
- Добавляем фото места
- Добавляем новых участников

Задачи

- Создать событие
- Добавить фото:
  - Обработать фото (сжатие, фильтры);
  - Отправить файл на сервер и получить URL;
  - Прикрепить к событию.
- Добавить участников

Схема задач

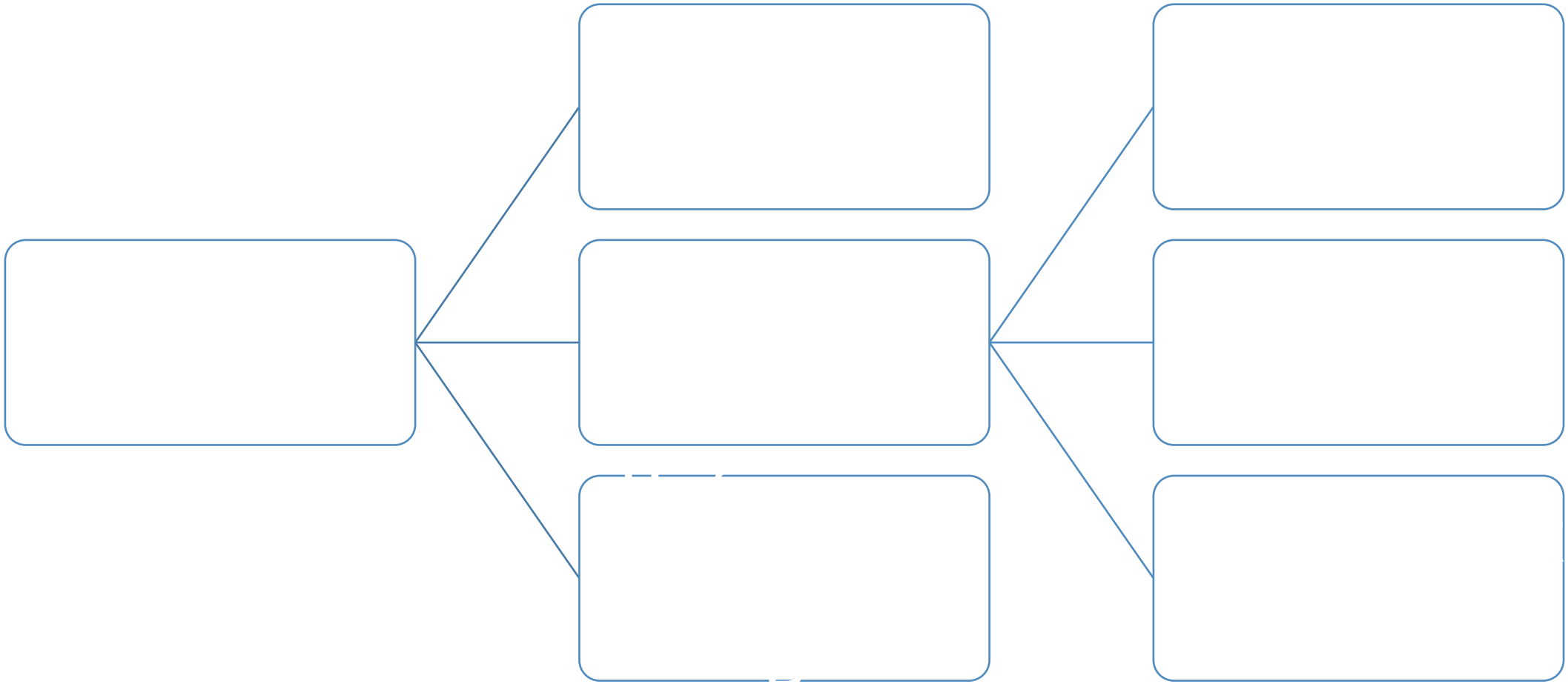


Схема задач

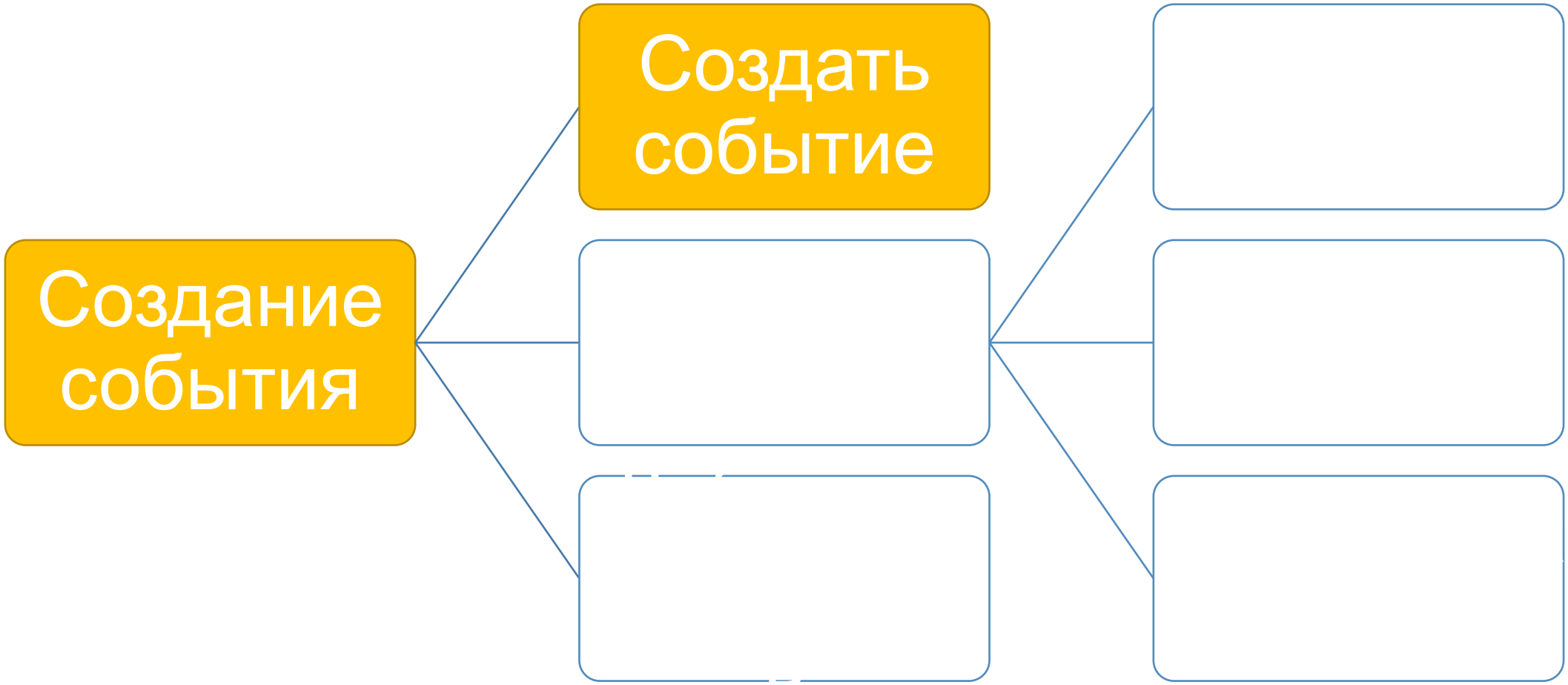


Схема задач

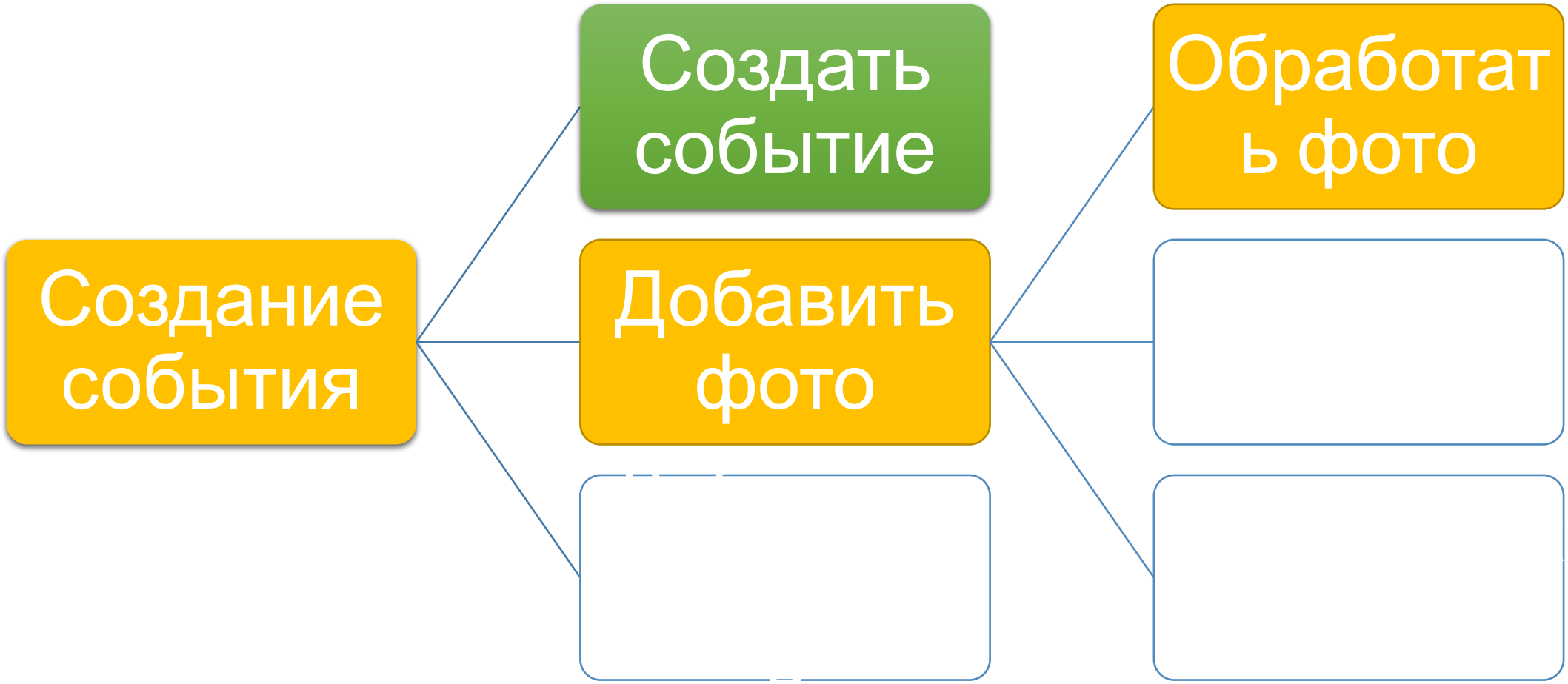

Создание события
Создать событие
Добавить фото
Обработать фото

Схема задач

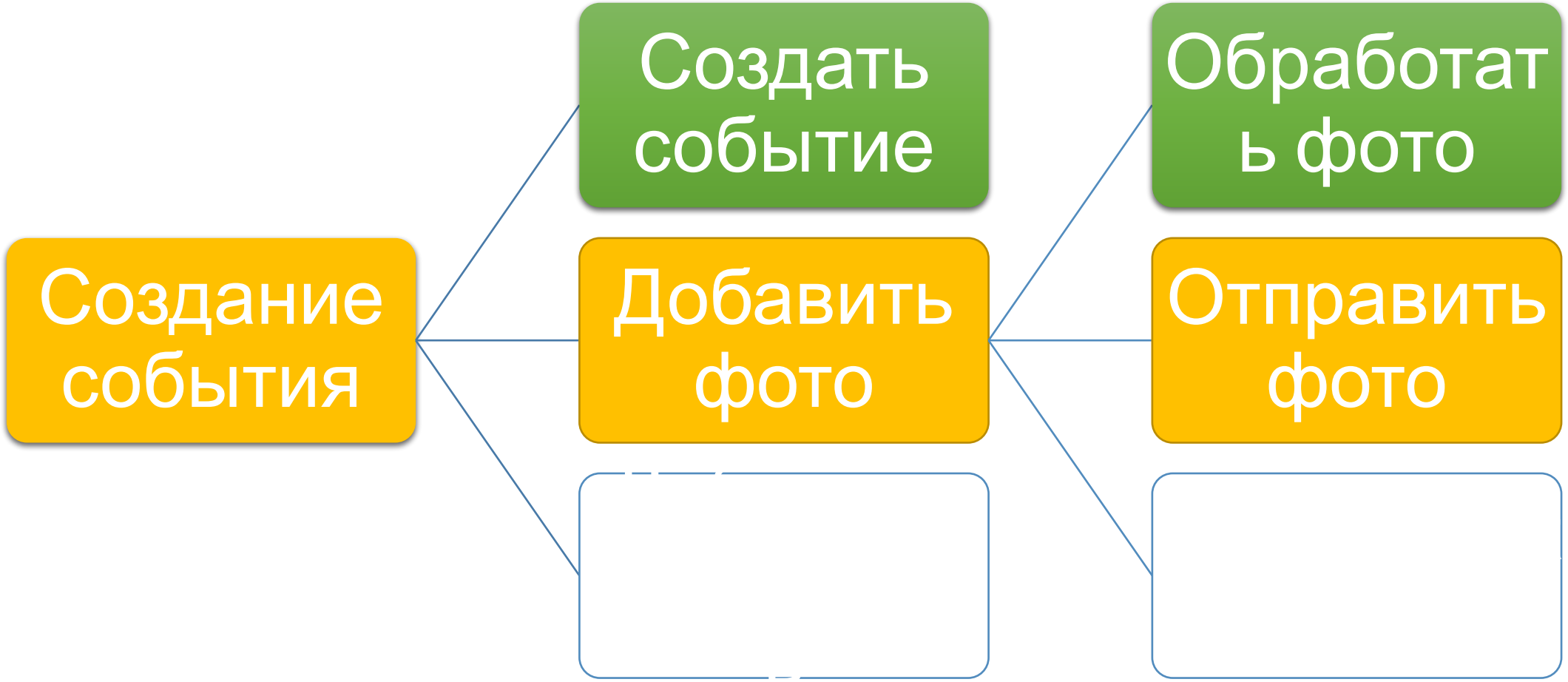

Создание события
Создать событие
Добавить фото
Обработать фото
Отправить фото

Схема задач

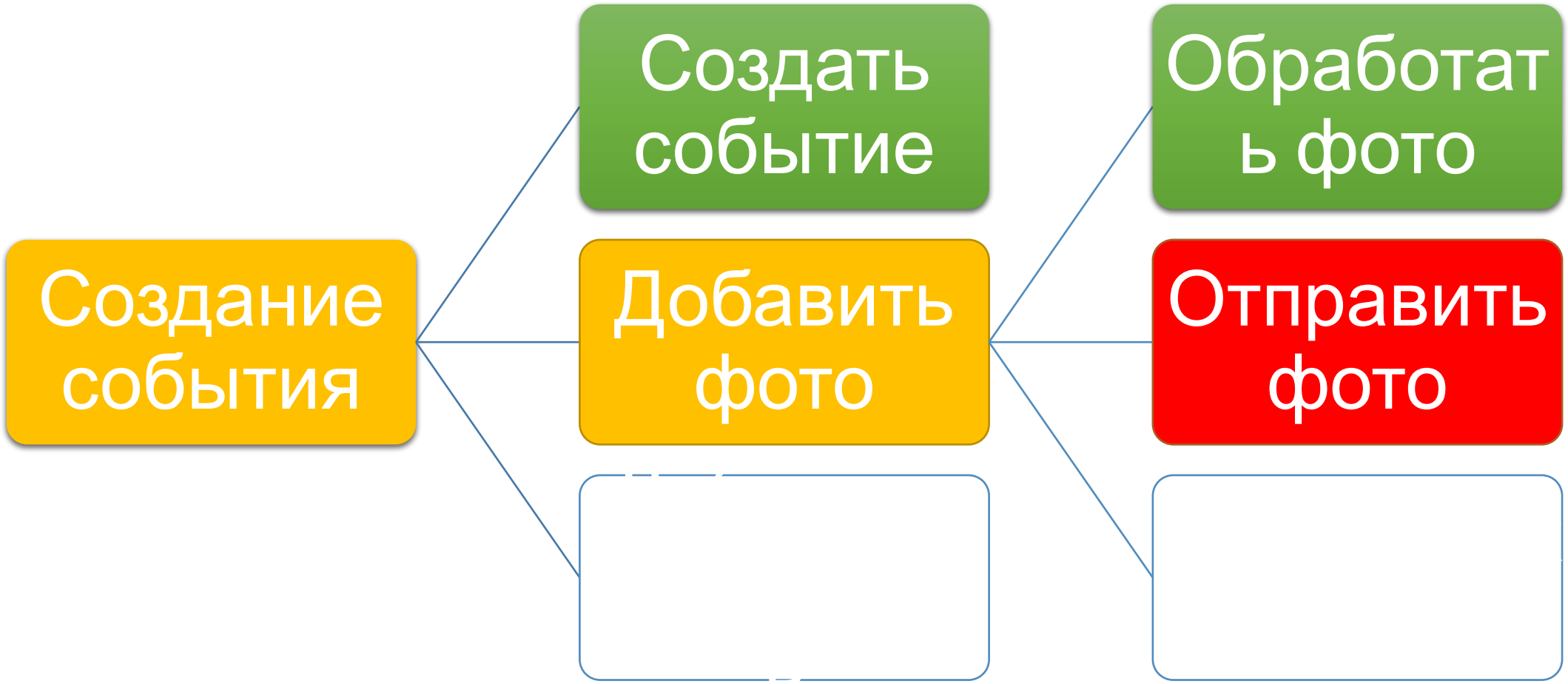

Создание события
Создать событие
Добавить фото
Обработать фото
Отправить фото

Схема задач

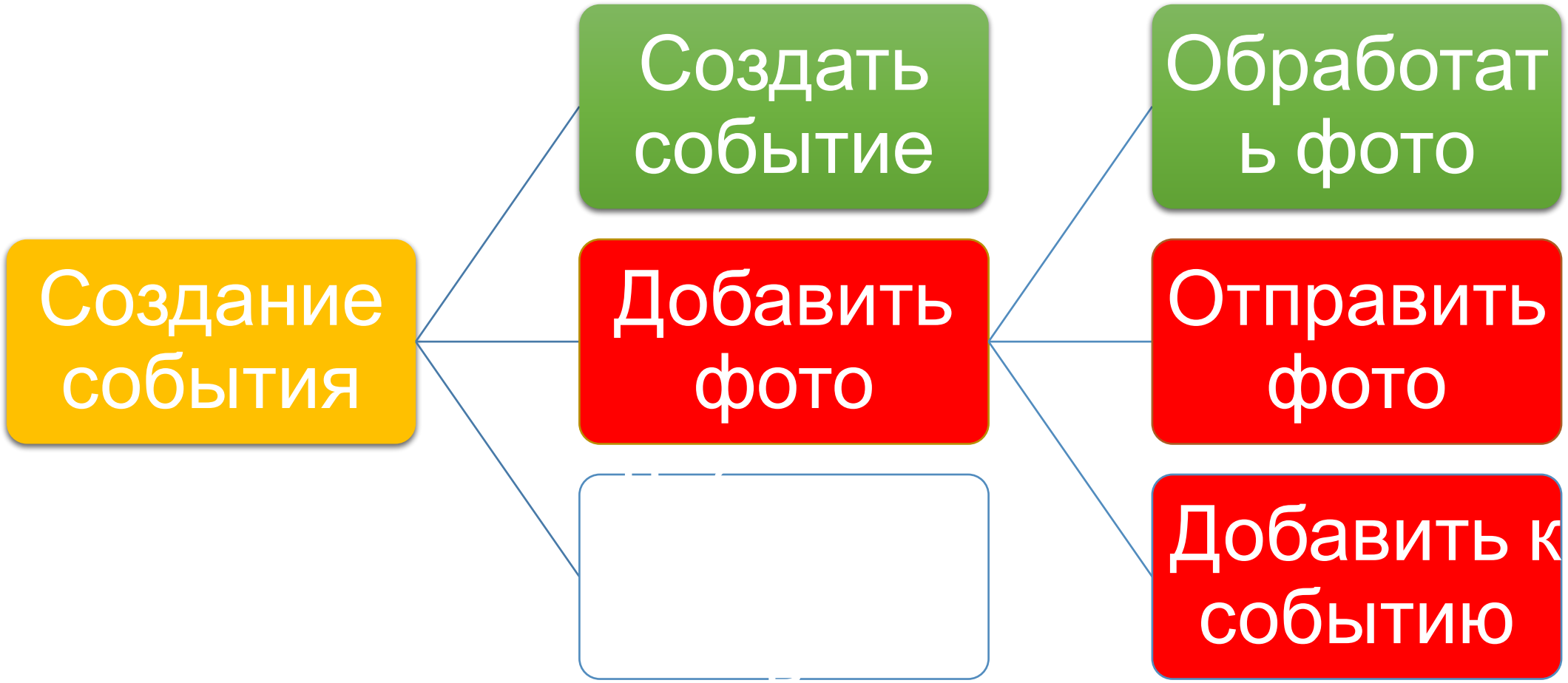

Создание события
Создать событие
Добавить фото
Обработать фото
Отправить фото
Добавить к событию

Схема задач

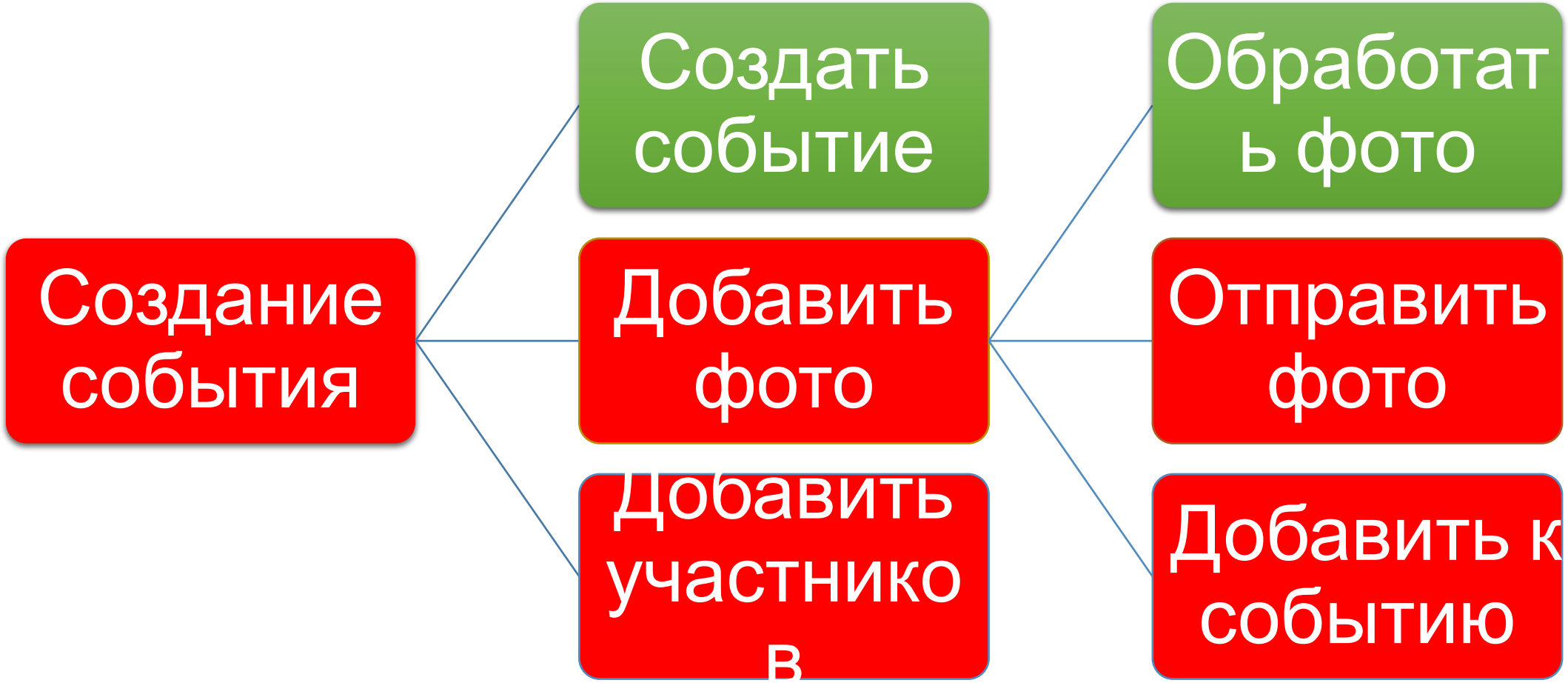

Создание события
Создать событие
Добавить фото
Добавить участников
Обработать фото
Отправить фото
Добавить к событию

Схема задач

Схема задач

Схема задач

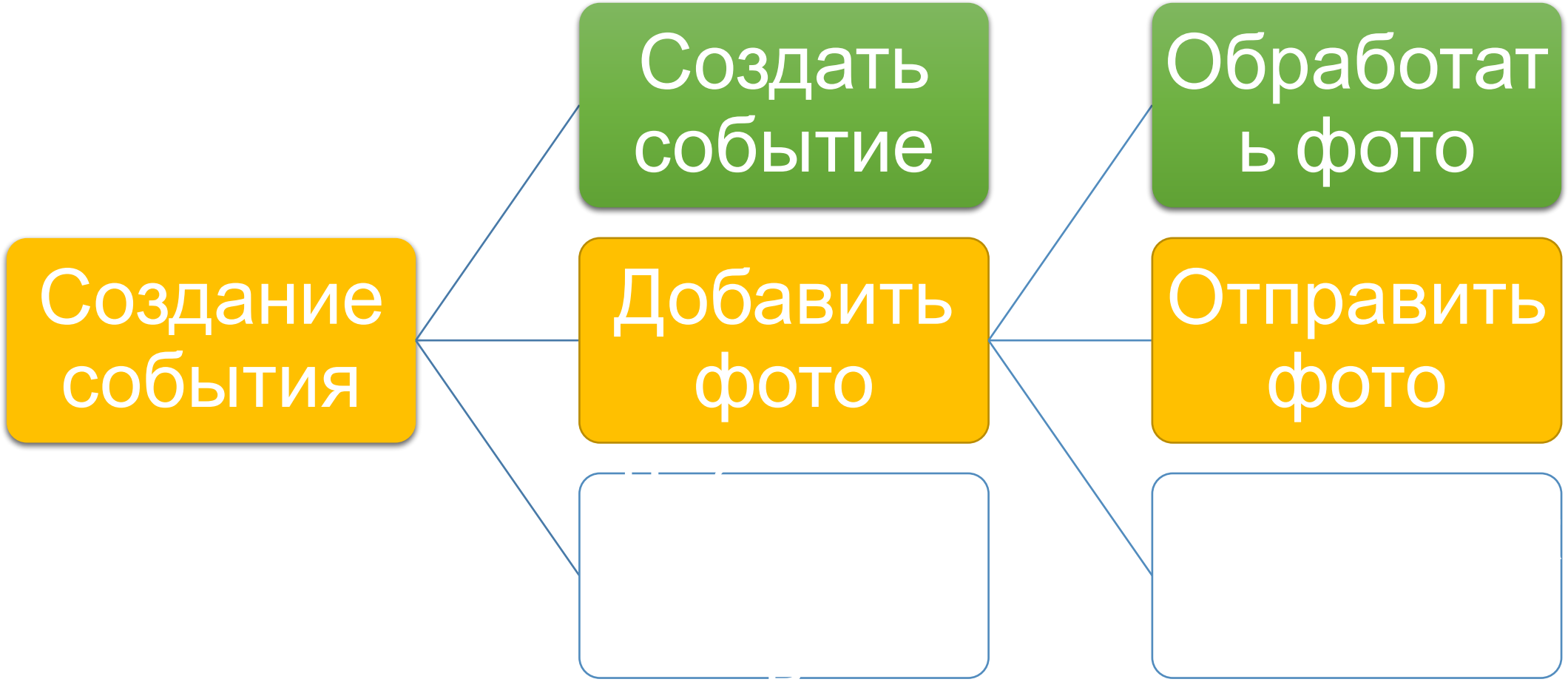

Создание события
Создать событие
Добавить фото
Обработать фото
Отправить фото

Схема задач

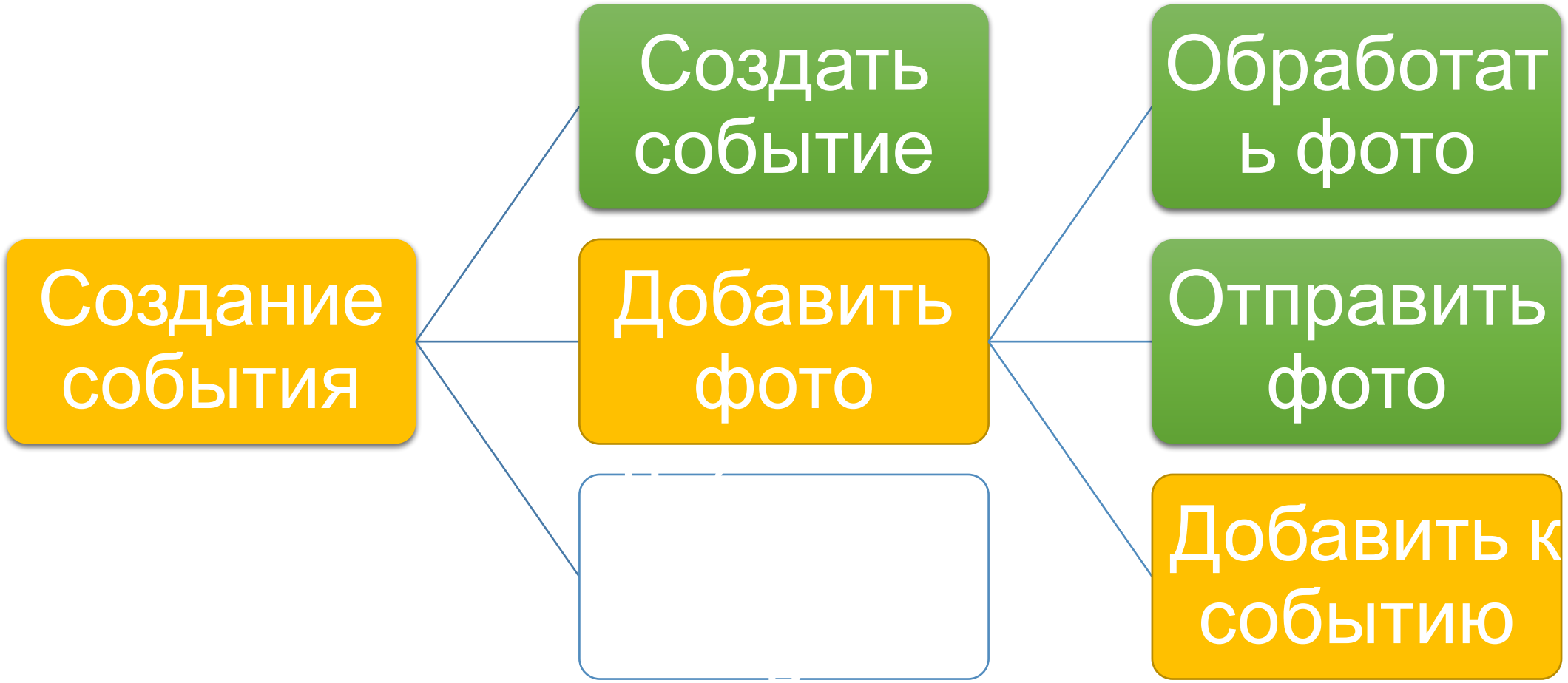

Создание события
Создать событие
Добавить фото
Обработать фото
Отправить фото
Добавить к событию

Схема задач

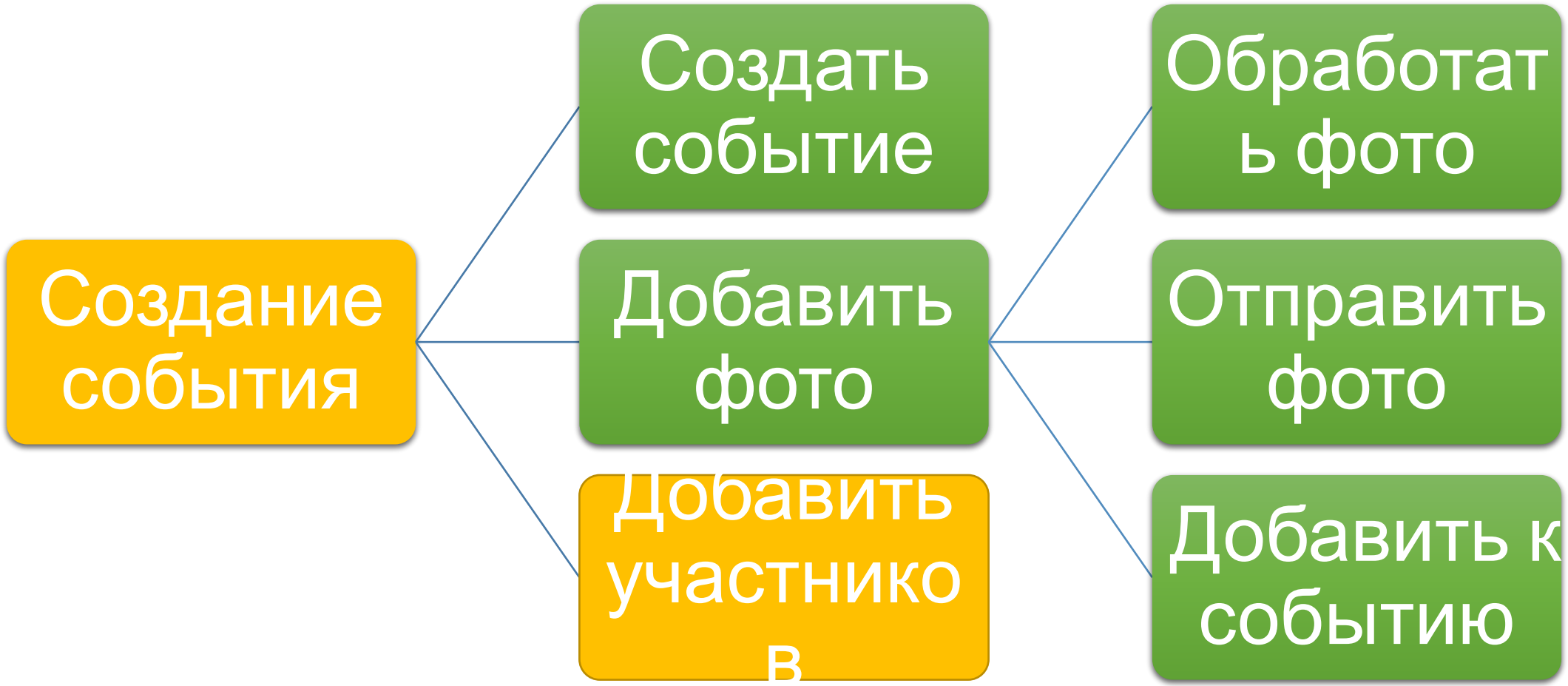
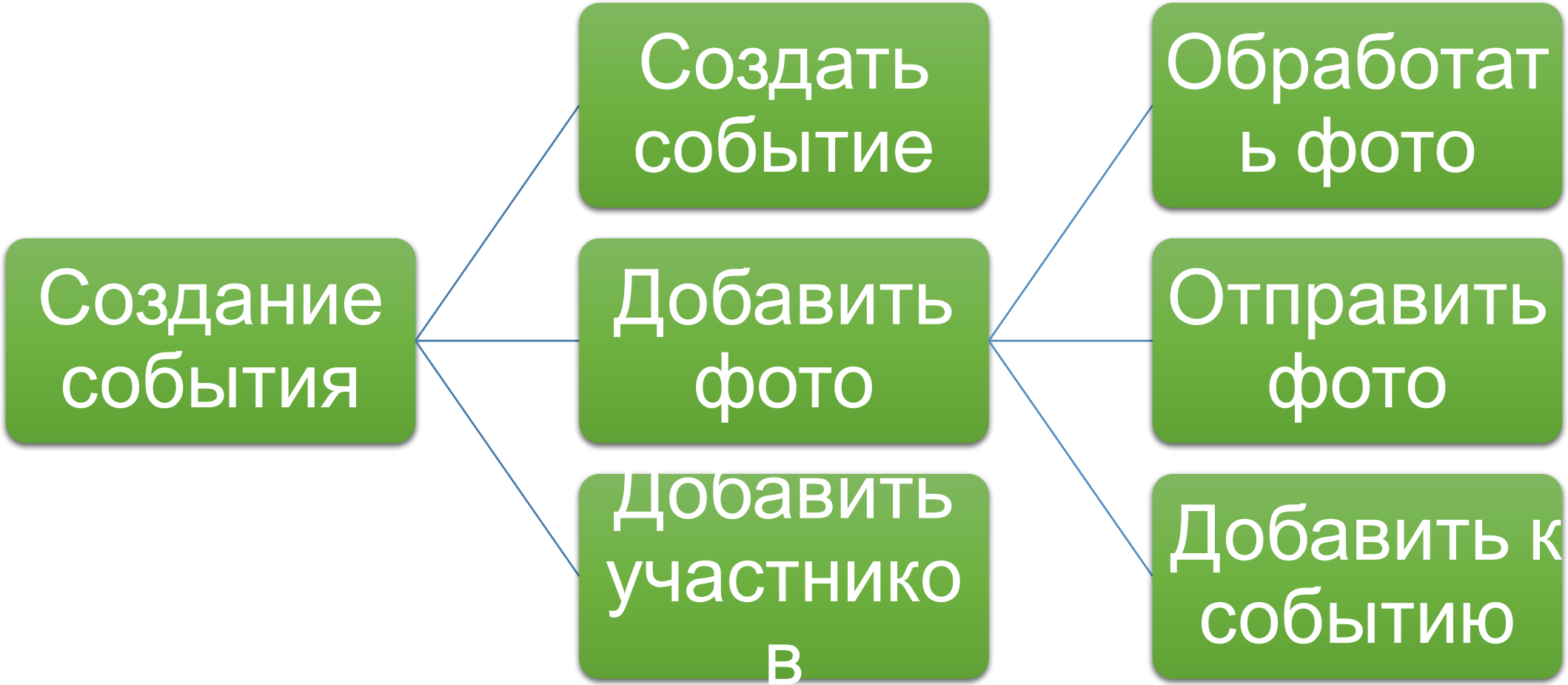

Создание события
Создать событие
Добавить фото
Добавить участников
Обработать фото
Отправить фото
Добавить к событию

Схема задач


Создание события
Создать событие
Добавить фото
Добавить участников
Обработать фото
Отправить фото
Добавить к событию

Отправка созданных пользователем данных

- Разбиваем задачи на атомарные подзадачи
- Сохранение данных и состояния задач
  - Семантически значимый ключ: EventID_TaskID_SubtaskID
  - Value в виде примитивных данных или сериализуемых структур, которые переживут изменения из-за обновлений сущностей

Запросы и их задачи

- Request = намерение
- RequestTask = как его достичь

Приложение для организации событий

- Создаём событие `CreateEventRequest`
- Добавляем фото места `AddPhotoRequest`
- Добавляем новых участников `AddParticipantRequest`

Приложение для организации событий

- Request -> RequestTask

Интерфейс RequestTask

```
public interface IRequestTask
{
  int Id { get; }
  Request Request { get; }
  ExecutingState State { get; }
  Task RunAsync();
  void Load(IReadOnlyDictionary<string, object> data);
  IReadOnlyDictionary<string, object> Save();
}
```

Интерфейс RequestTask

```
public interface IRequestTask
{
  int Id { get; }
  Request Request { get; }
  ExecutingState State { get; }
  Task RunAsync();
  void Load(IReadOnlyDictionary<string, object> data);
  IReadOnlyDictionary<string, object> Save();
}
```

Интерфейс RequestTask

```
public interface IRequestTask
{
  int Id { get; }
  Request Request { get; }
  ExecutingState State { get; }
  Task RunAsync();
  void Load(IReadOnlyDictionary<string, object> data);
  IReadOnlyDictionary<string, object> Save();
}
```

Интерфейс RequestTask

```
public interface IRequestTask
{
  int Id { get; }
  Request Request { get; }
  ExecutingState State { get; }
  Task RunAsync();
  void Load(IReadOnlyDictionary<string, object> data);
  IReadOnlyDictionary<string, object> Save();
}
```

Интерфейс RequestTask

```
public interface IRequestTask
{
  int Id { get; }
  Request Request { get; }
  ExecutingState State { get; }
  Task RunAsync();
  void Load(IReadOnlyDictionary<string, object> data);
  IReadOnlyDictionary<string, object> Save();
}
```

Интерфейс RequestTask

```
public interface IRequestTask
{
  int Id { get; }
  Request Request { get; }
  ExecutingState State { get; }
  Task RunAsync();
  void Load(IReadOnlyDictionary<string, object> data);
  IReadOnlyDictionary<string, object> Save();
}
```

Интерфейс RequestTask

```
public interface IRequestTask
{
   int Id { get; }
   Request Request { get; }
   ExecutingState State { get; }
   Task RunAsync();
   void Load(IReadOnlyDictionary<string, object> data);
   IReadOnlyDictionary<string, object> Save();
}
```

Приложение для организации событий

- CreateEventRequest CreateEventRequestTask
- AddPhotoRequest
  CompressPhotoRequestTask
  SendPhotoRequestTask
  AddPhotoToEventRequestTask
- AddParticipantRequest AddParticipantRequestTask

Очередь

- CreateEvent
- CompressPhoto
- SendPhoto
- AddPhotoToEvent
- AddParticipants

Очередь

- CreateEvent
- CompressPhoto
- SendPhoto
- AddPhotoToEvent
- AddParticipants

Executor 1

Executor 2

SerialRequestTask

```
public class SerialRequestTask : IRequestTask
{
  private readonly IRequestTask _wrappedTask;
  private readonly IRequestTask _dependingOnTask;
  …
}
```

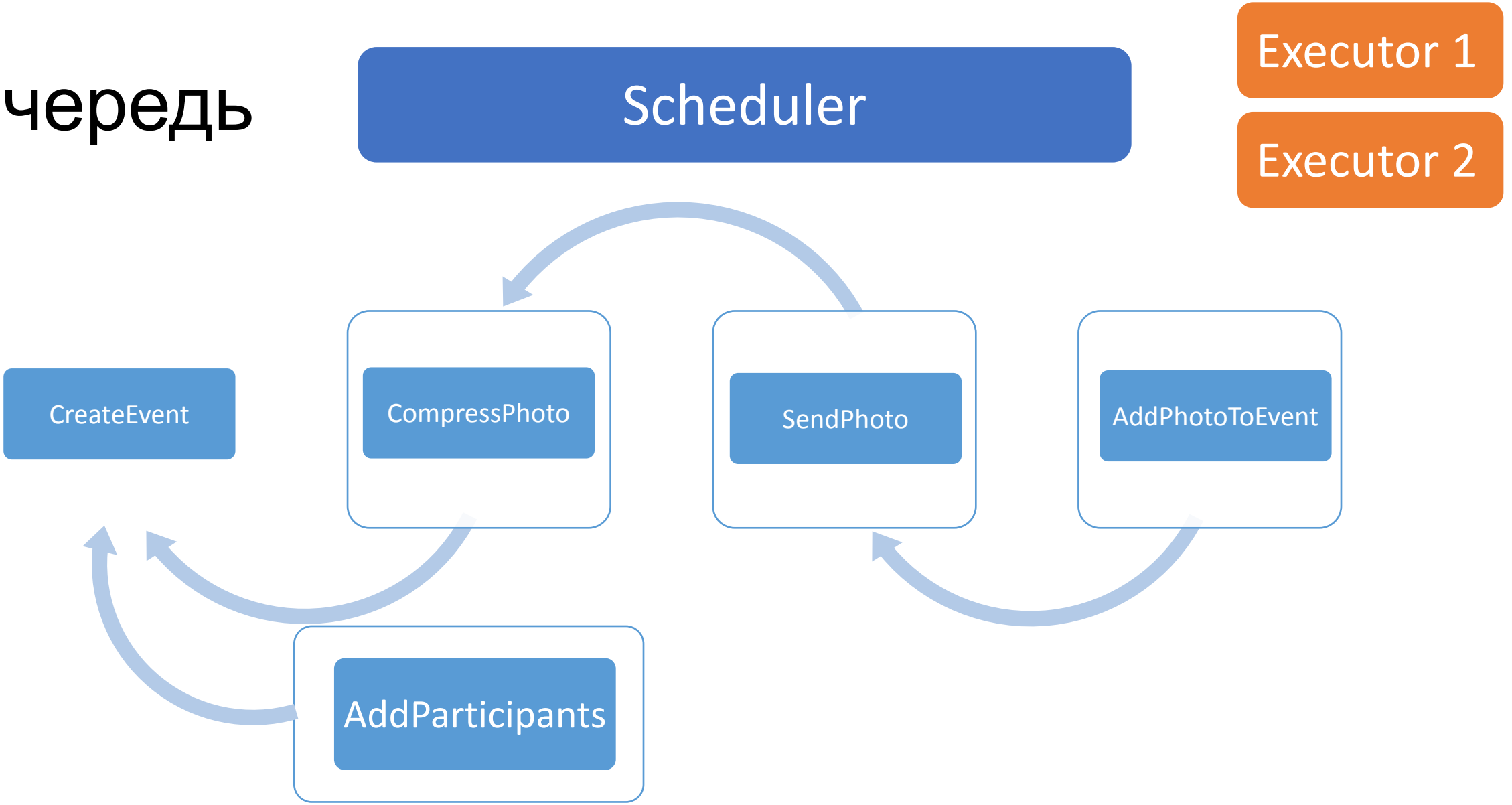
SerialRequestTask

```
public class SerialRequestTask : IRequestTask
{
  private readonly IRequestTask _wrappedTask;
  private readonly IRequestTask _dependingOnTask;
  …
}
```

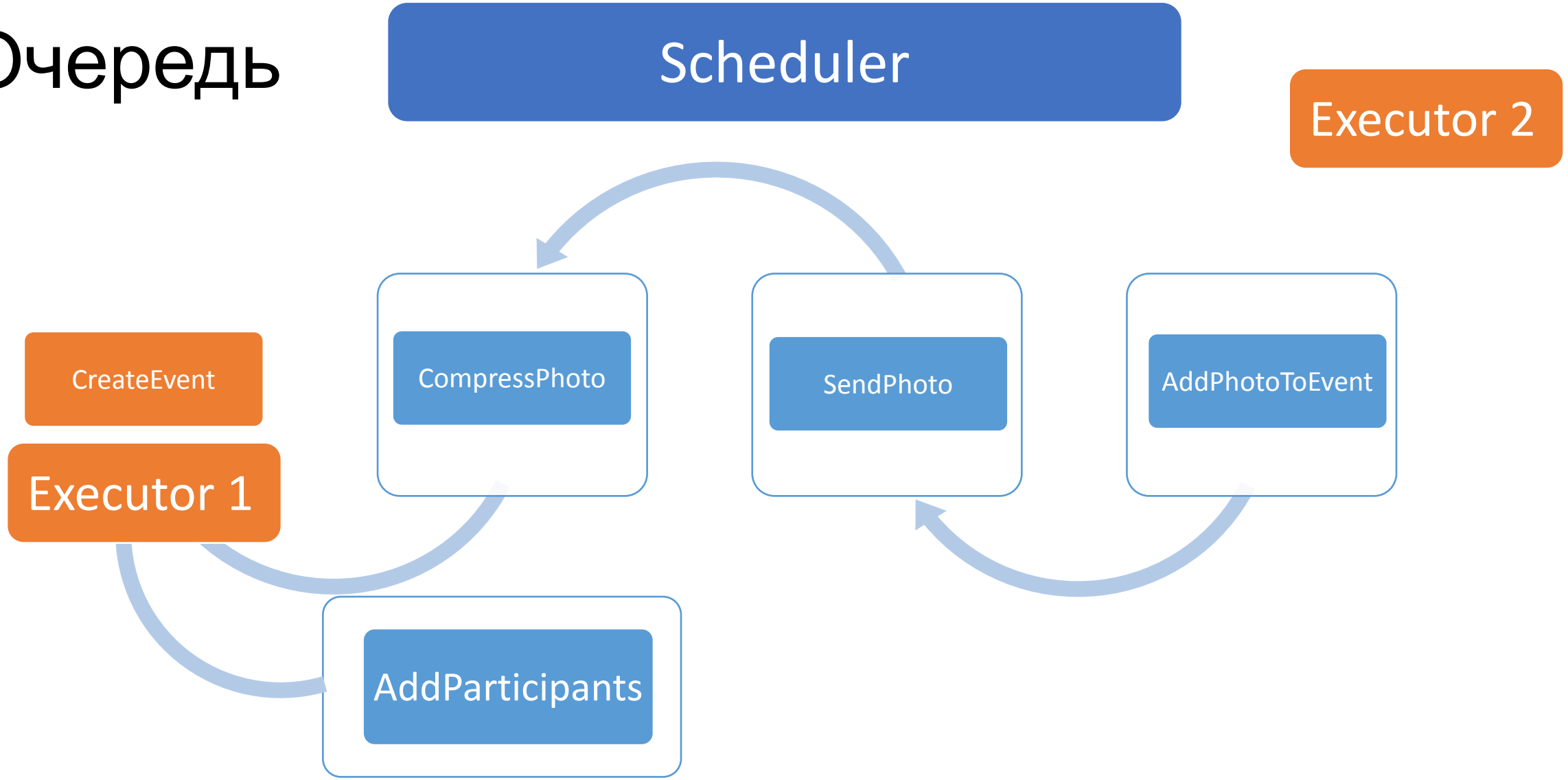
SerialRequestTask

```
public class SerialRequestTask : IRequestTask
{
  private readonly IRequestTask _wrappedTask;
  private readonly IRequestTask _dependingOnTask;
  …
}
```

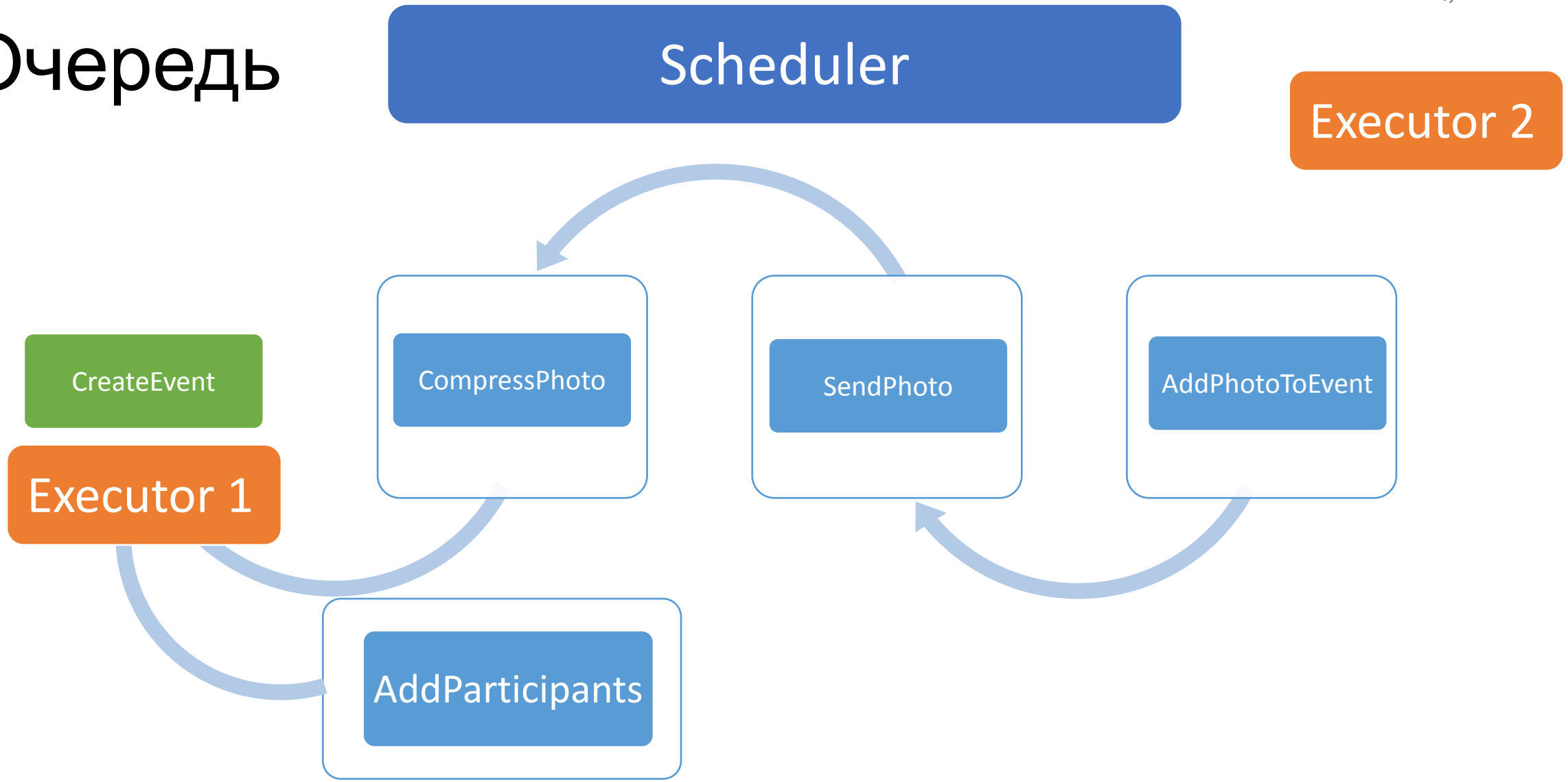
Очередь


Scheduler
Executor 1
Executor 2
CreateEvent
CompressPhoto
SendPhoto
AddPhotoToEvent
AddParticipants

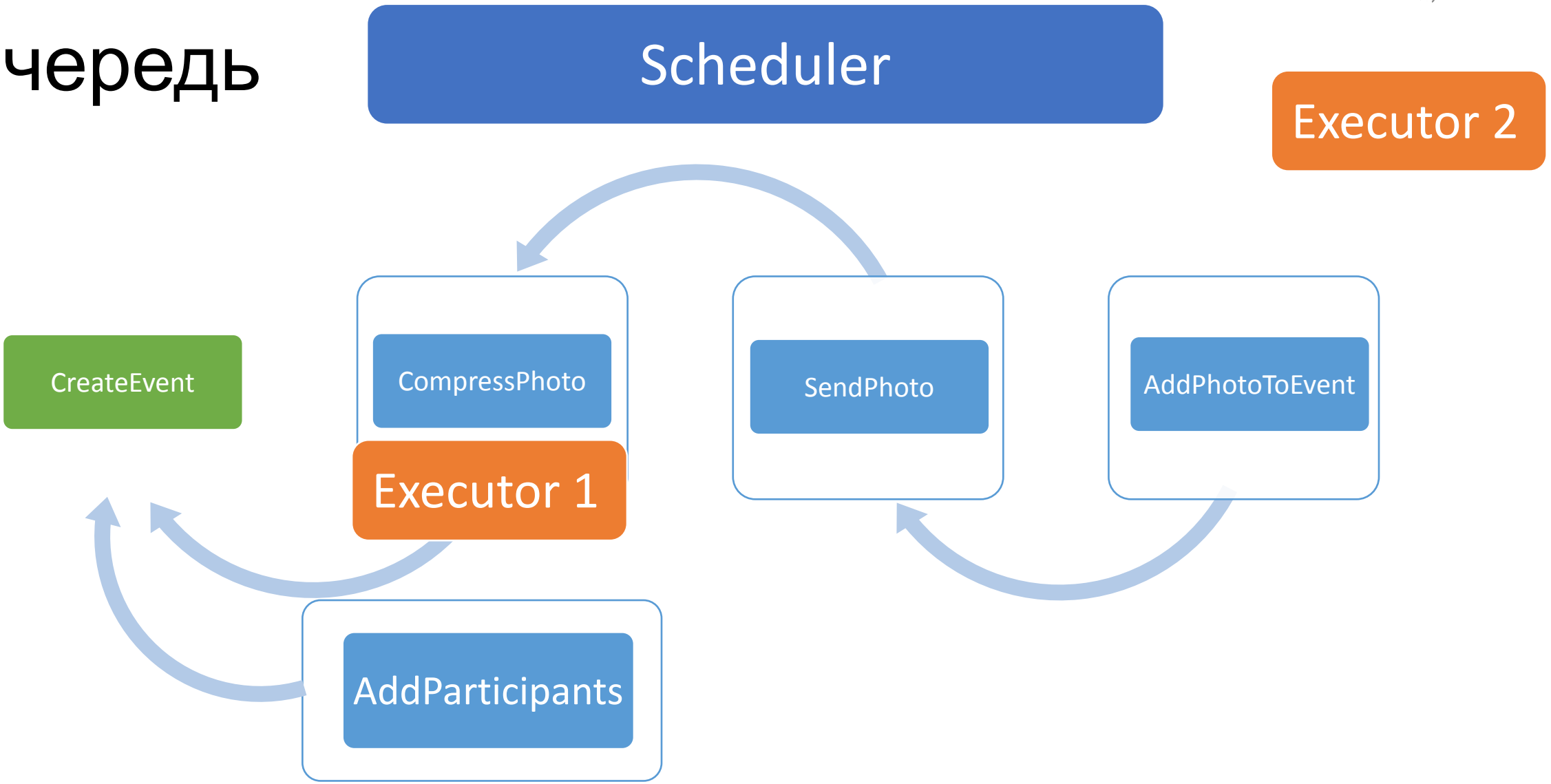
Очередь


Scheduler
Executor 2
CreateEvent
Executor 1
CompressPhoto
SendPhoto
AddPhotoToEvent
AddParticipants

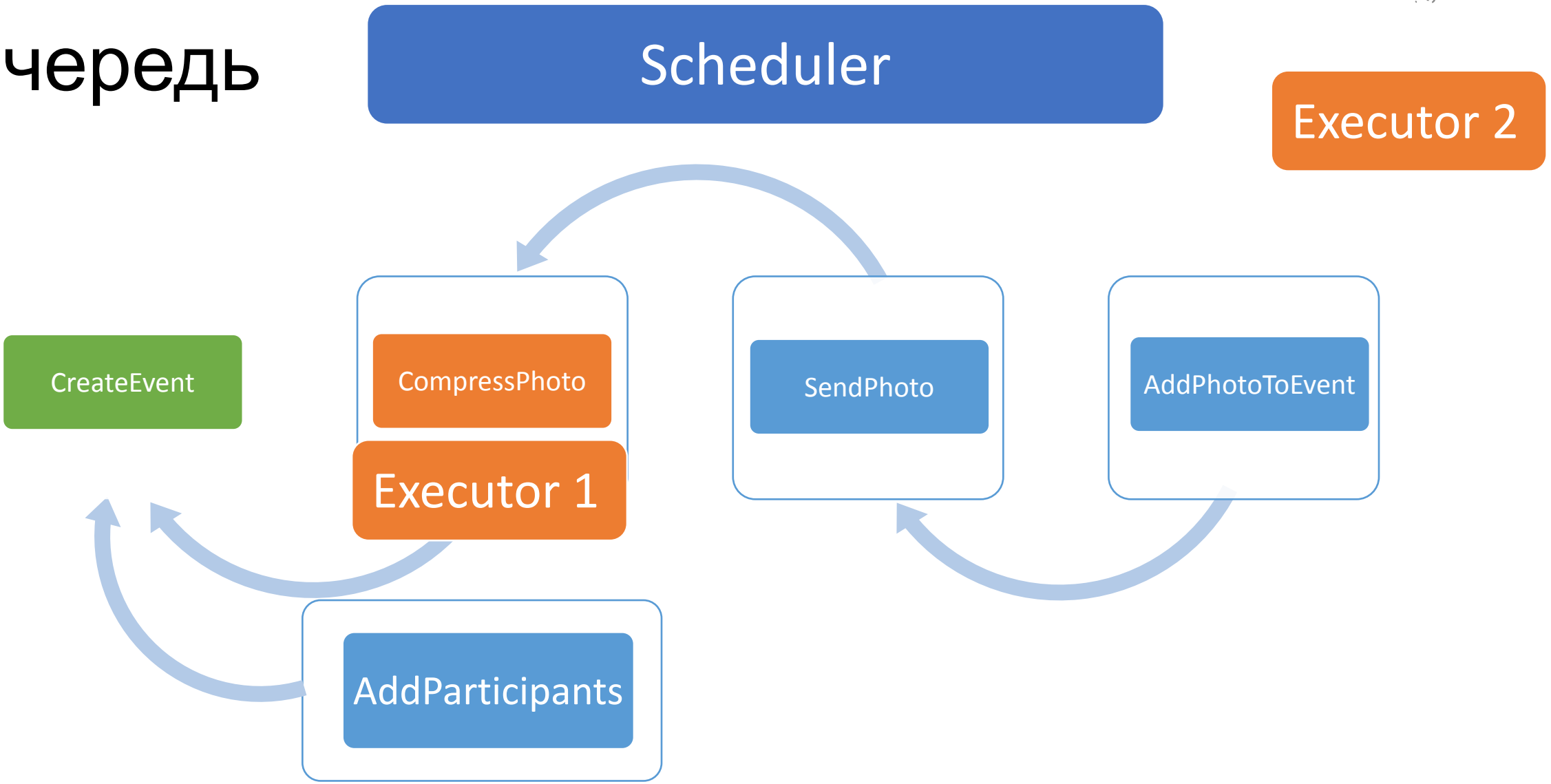
Очередь


Scheduler
Executor 2
CreateEvent
Executor 1
CompressPhoto
SendPhoto
AddPhotoToEvent
AddParticipants

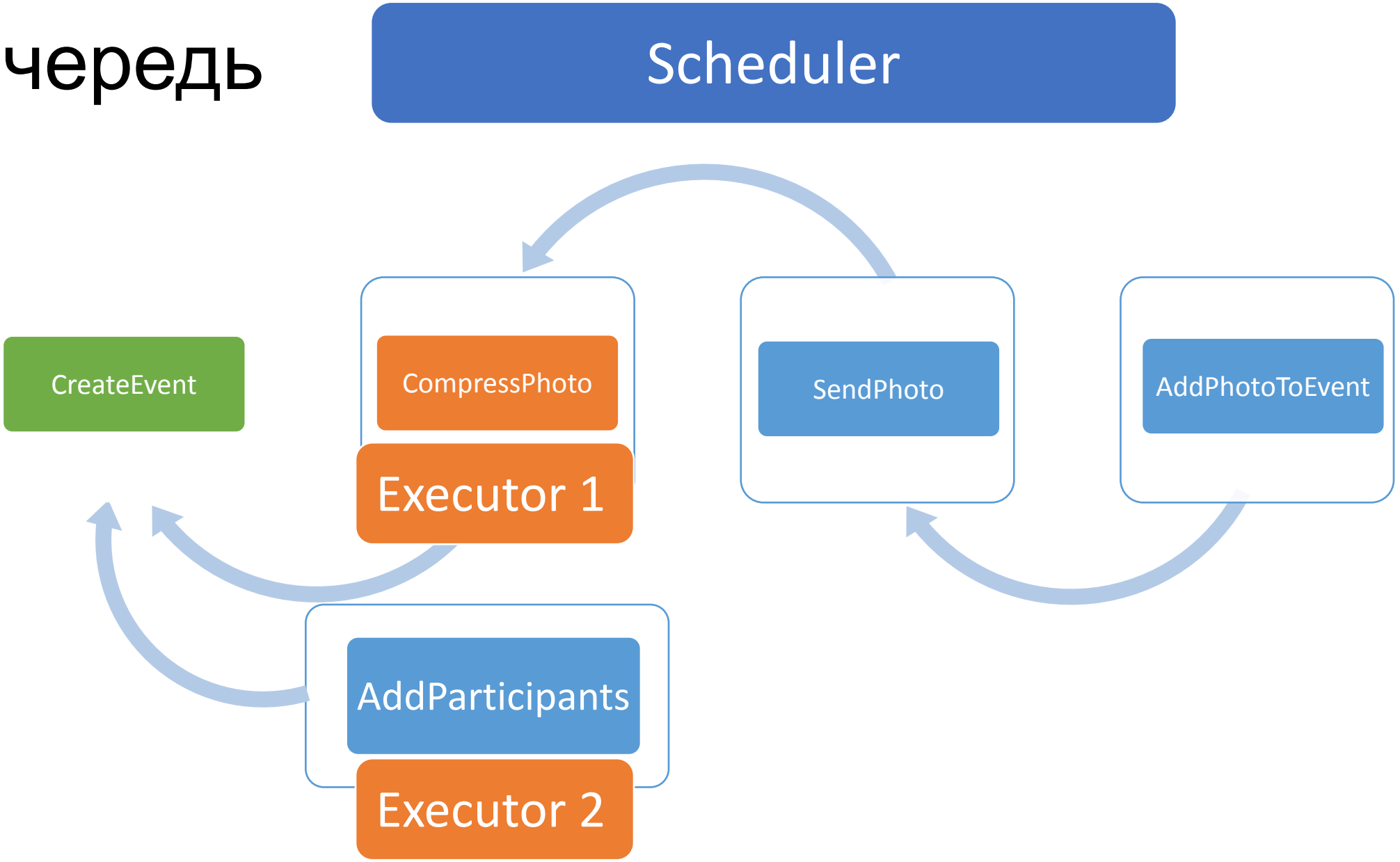
Очередь


Scheduler
Executor 2
CreateEvent
CompressPhoto
Executor 1
SendPhoto
AddPhotoToEvent
AddParticipants

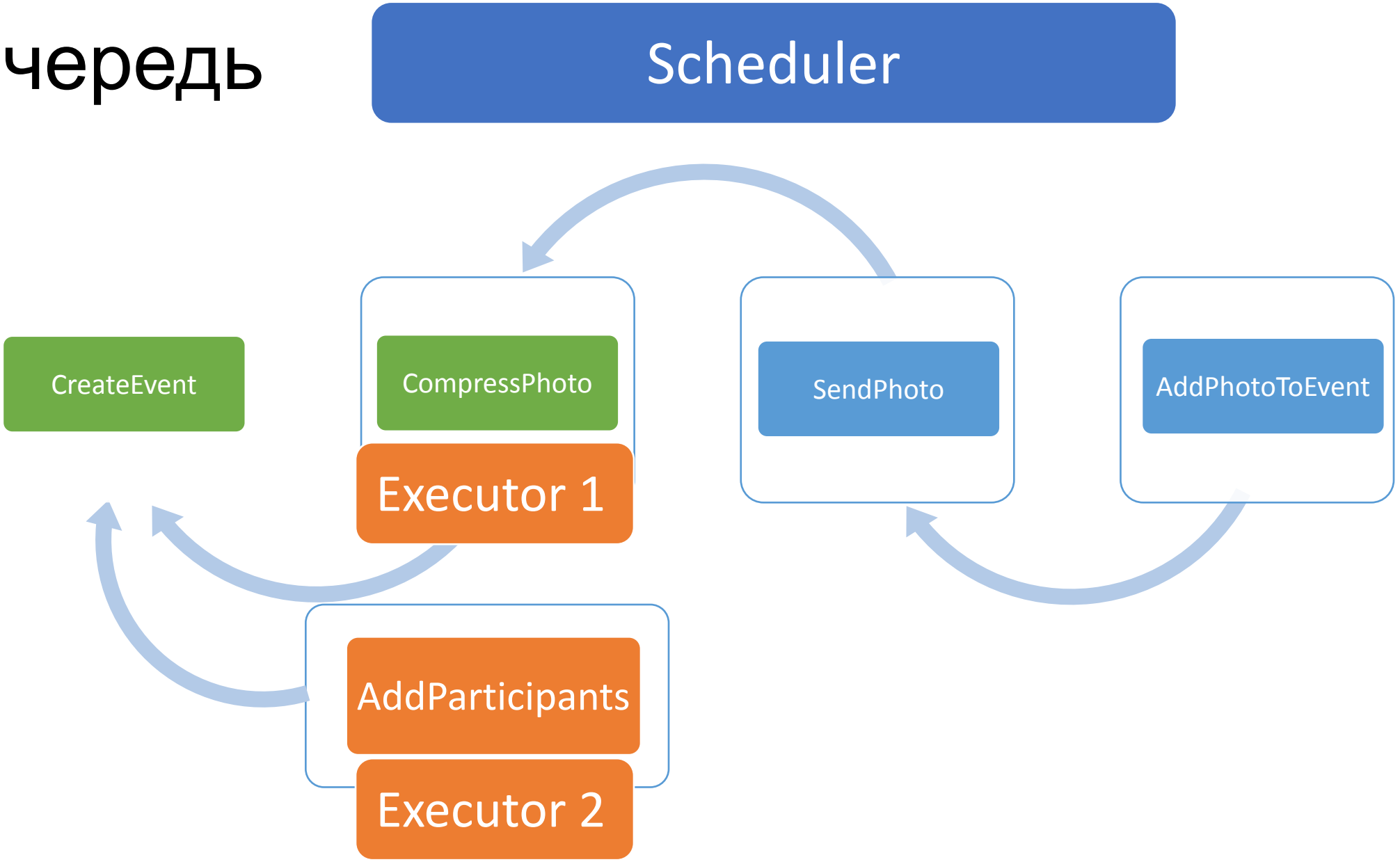
Очередь


Scheduler
Executor 2
CreateEvent
CompressPhoto
Executor 1
SendPhoto
AddPhotoToEvent
AddParticipants

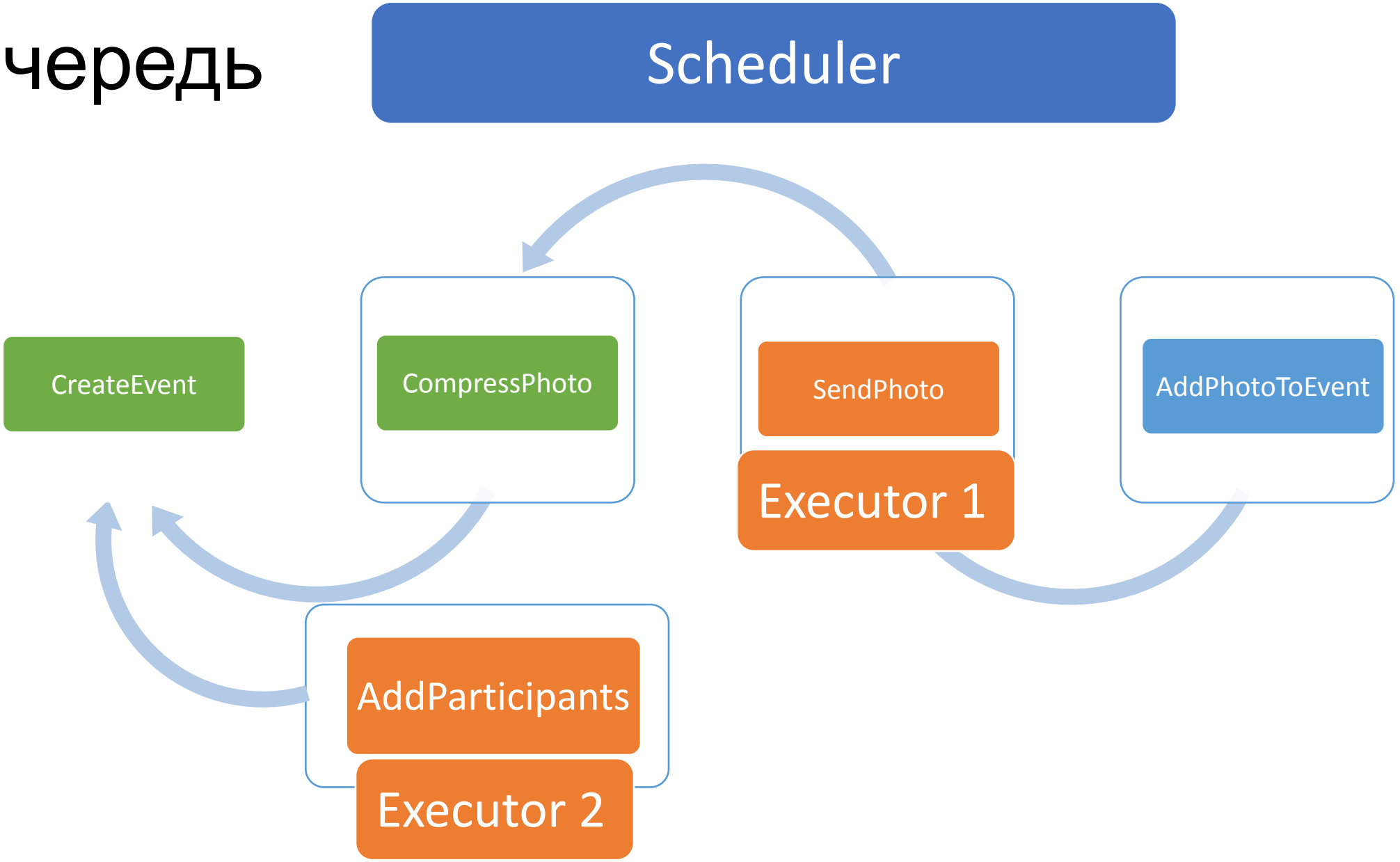
Очередь


Scheduler
CreateEvent
CompressPhoto
Executor 1
SendPhoto
AddPhotoToEvent
AddParticipants
Executor 2

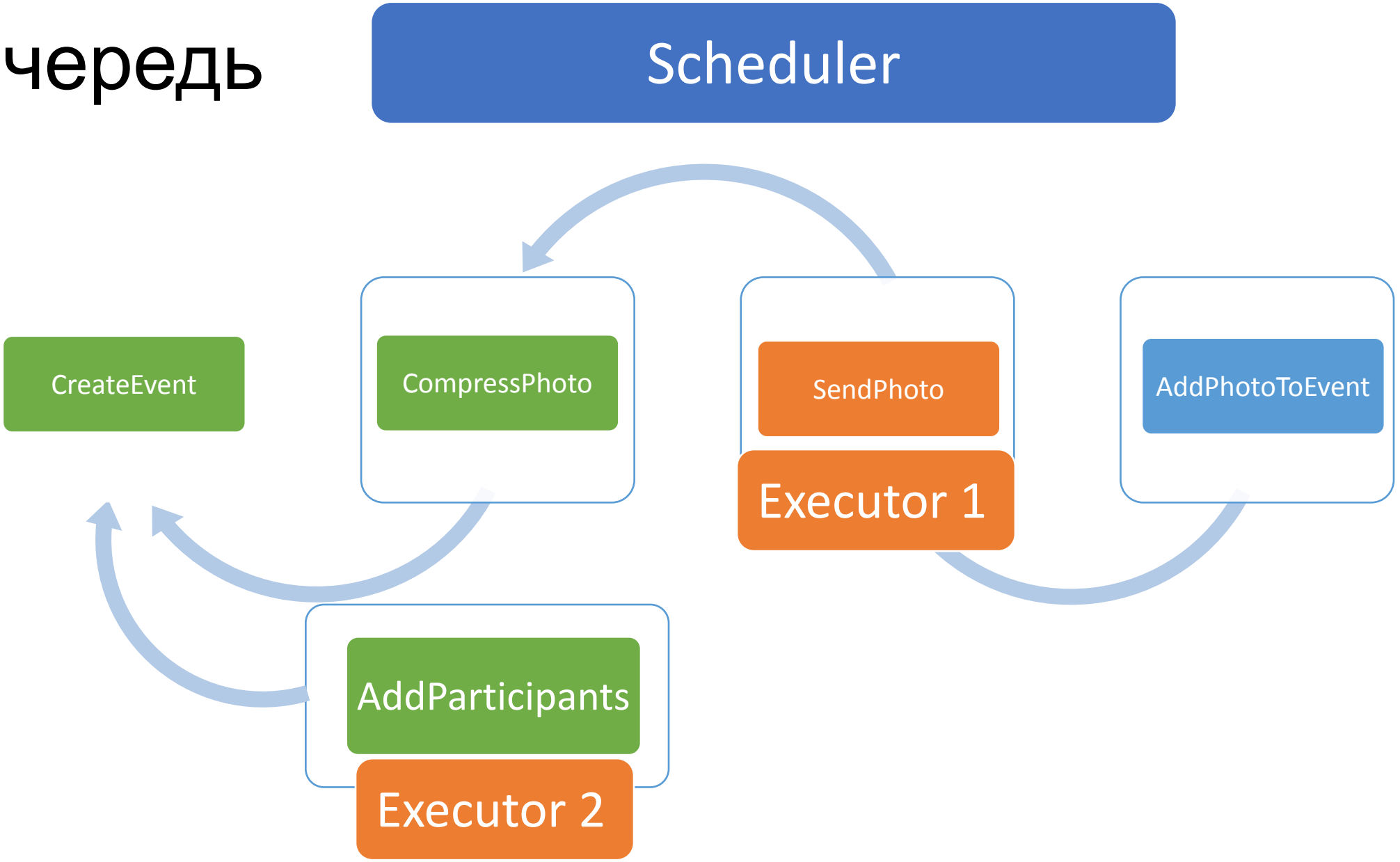
Очередь


Scheduler
CreateEvent
CompressPhoto
Executor 1
SendPhoto
AddPhotoToEvent
AddParticipants
Executor 2

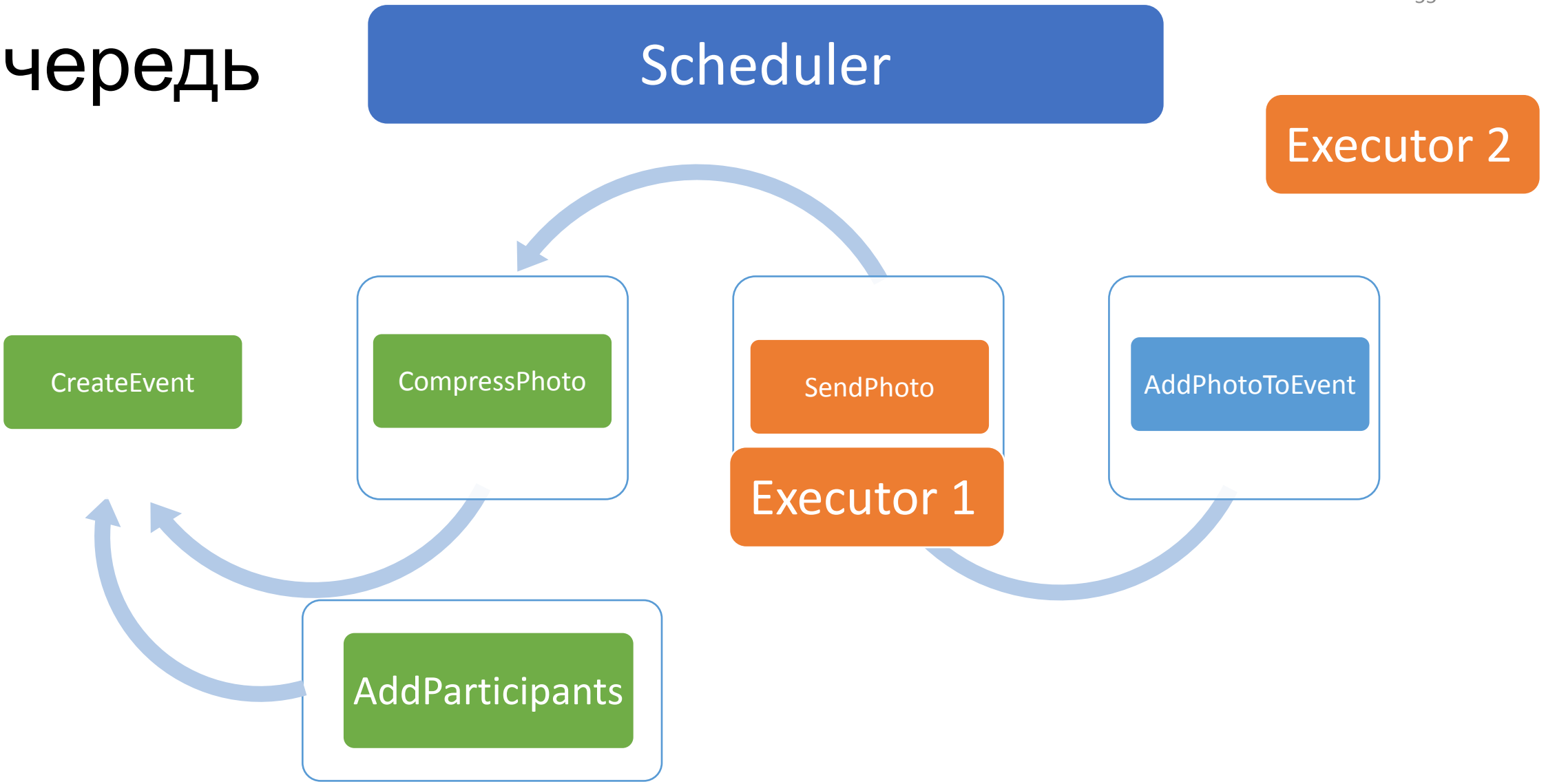
Очередь


Scheduler
CreateEvent
CompressPhoto
SendPhoto
Executor 1
AddPhotoToEvent
AddParticipants
Executor 2

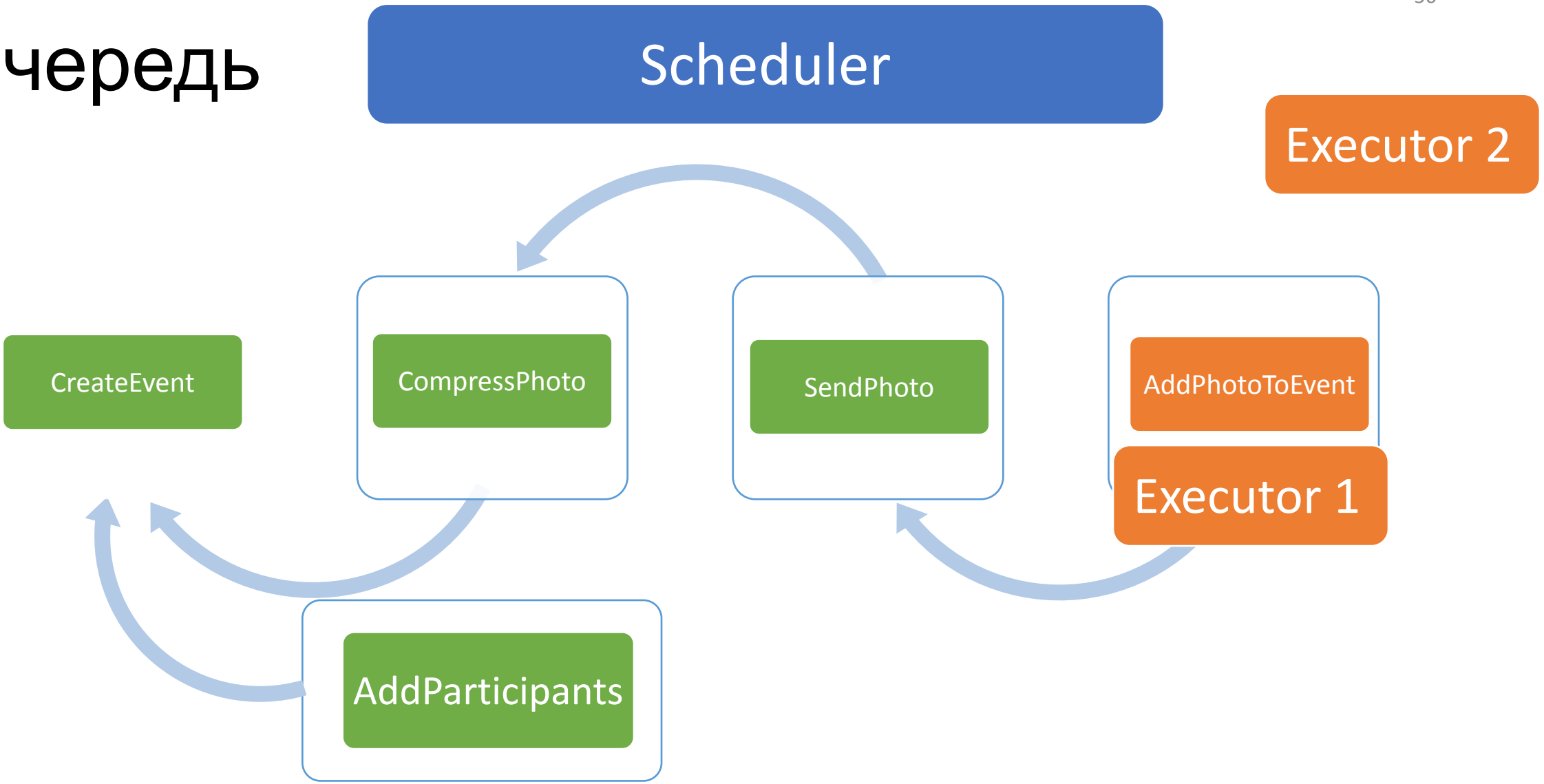
Очередь


Scheduler
CreateEvent
CompressPhoto
SendPhoto
Executor 1
AddPhotoToEvent
AddParticipants
Executor 2

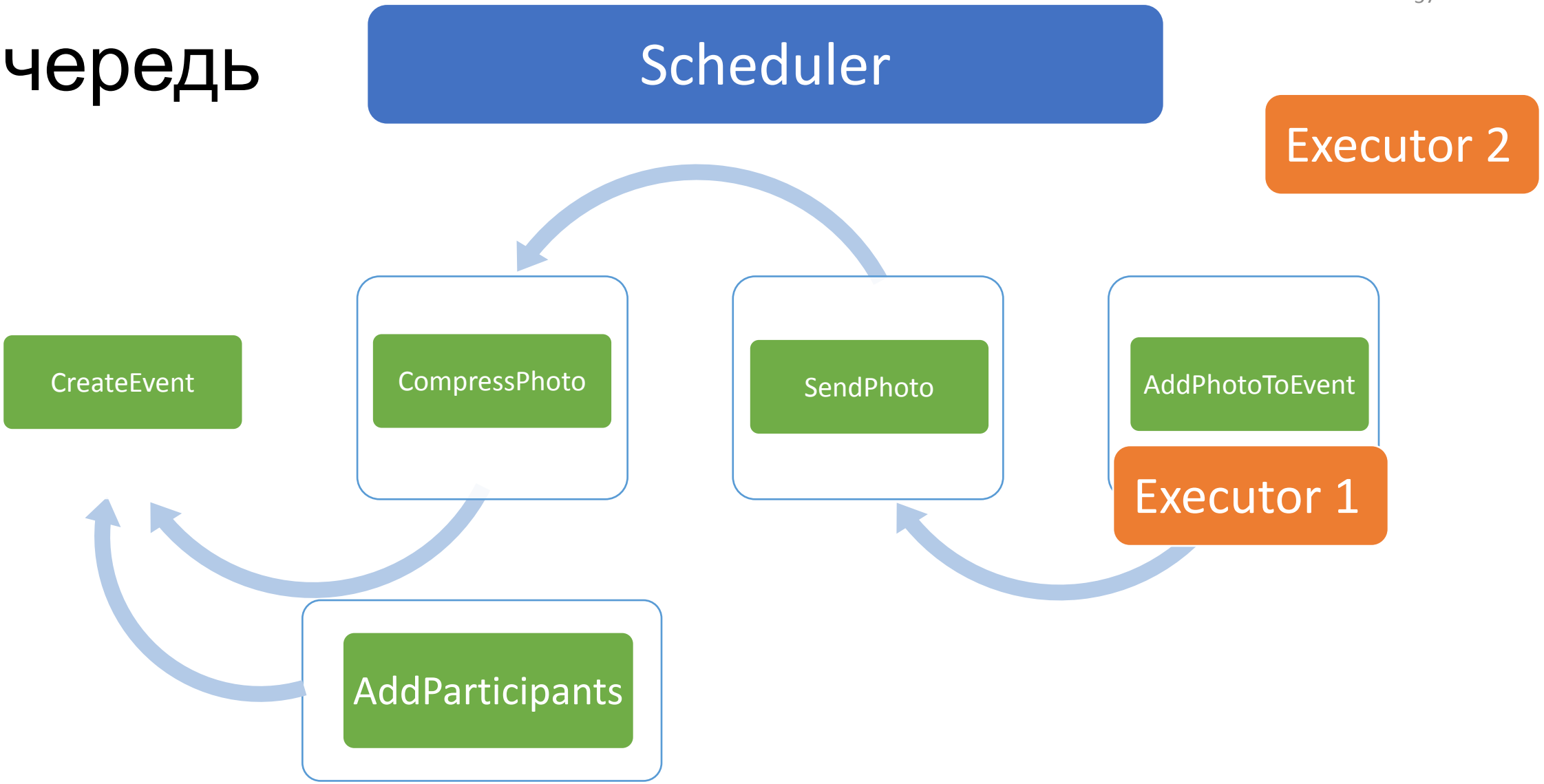
Очередь


Scheduler
Executor 2
CreateEvent
CompressPhoto
SendPhoto
AddPhotoToEvent
Executor 1
AddParticipants

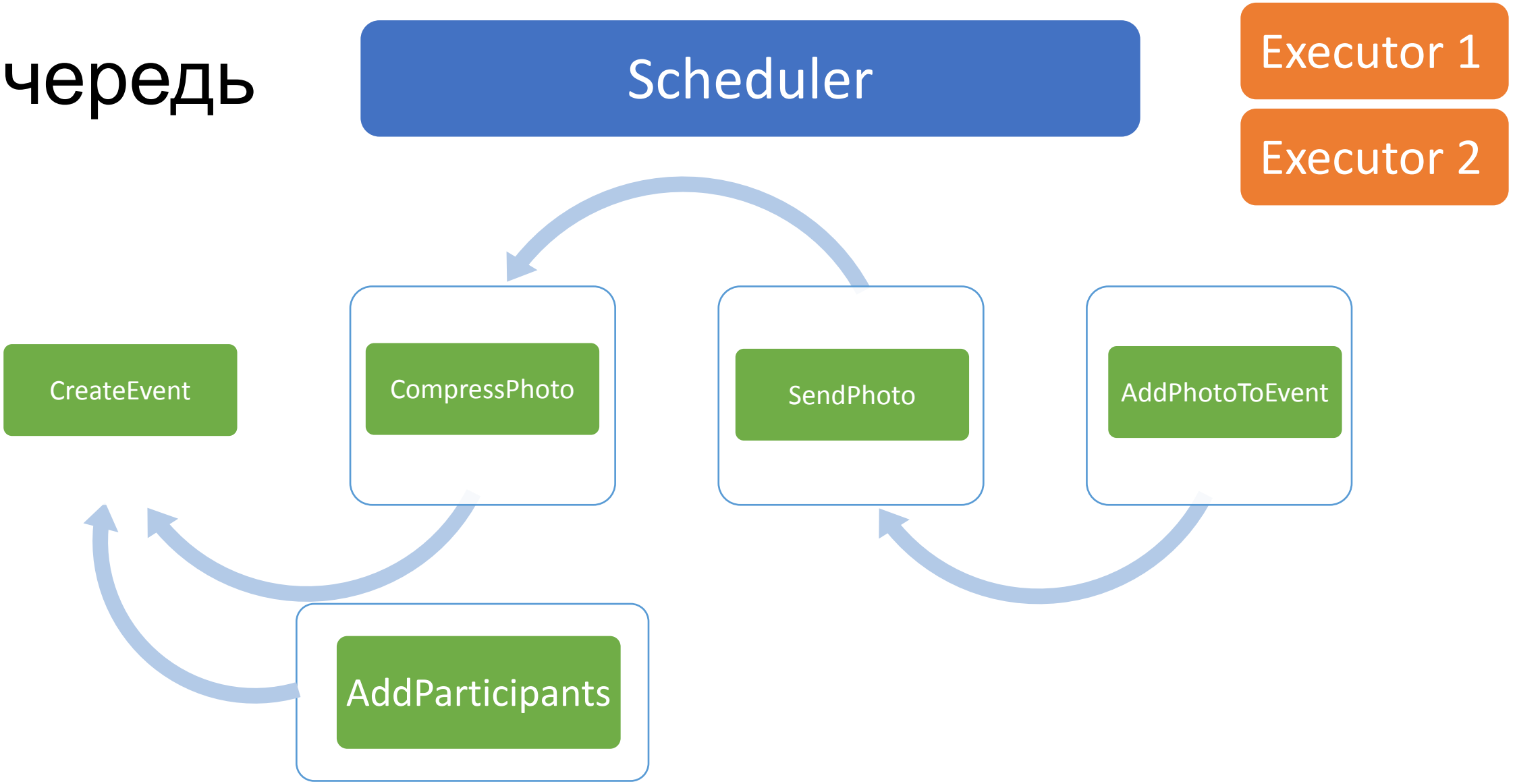
Очередь


Scheduler
Executor 2
CreateEvent
CompressPhoto
SendPhoto
AddPhotoToEvent
Executor 1
AddParticipants

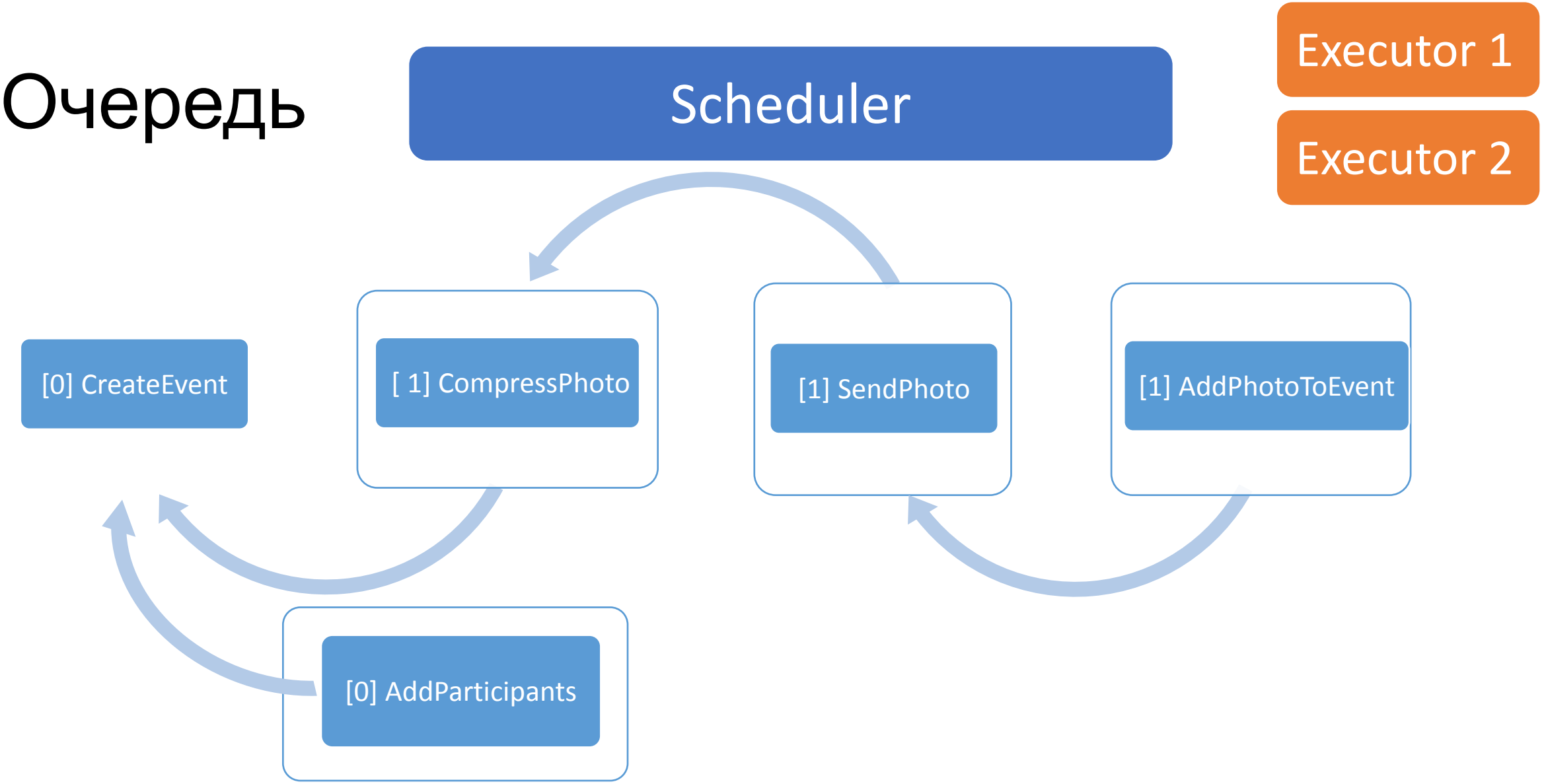
Очередь


Scheduler
Executor 2
CreateEvent
CompressPhoto
SendPhoto
AddPhotoToEvent
Executor 1
AddParticipants

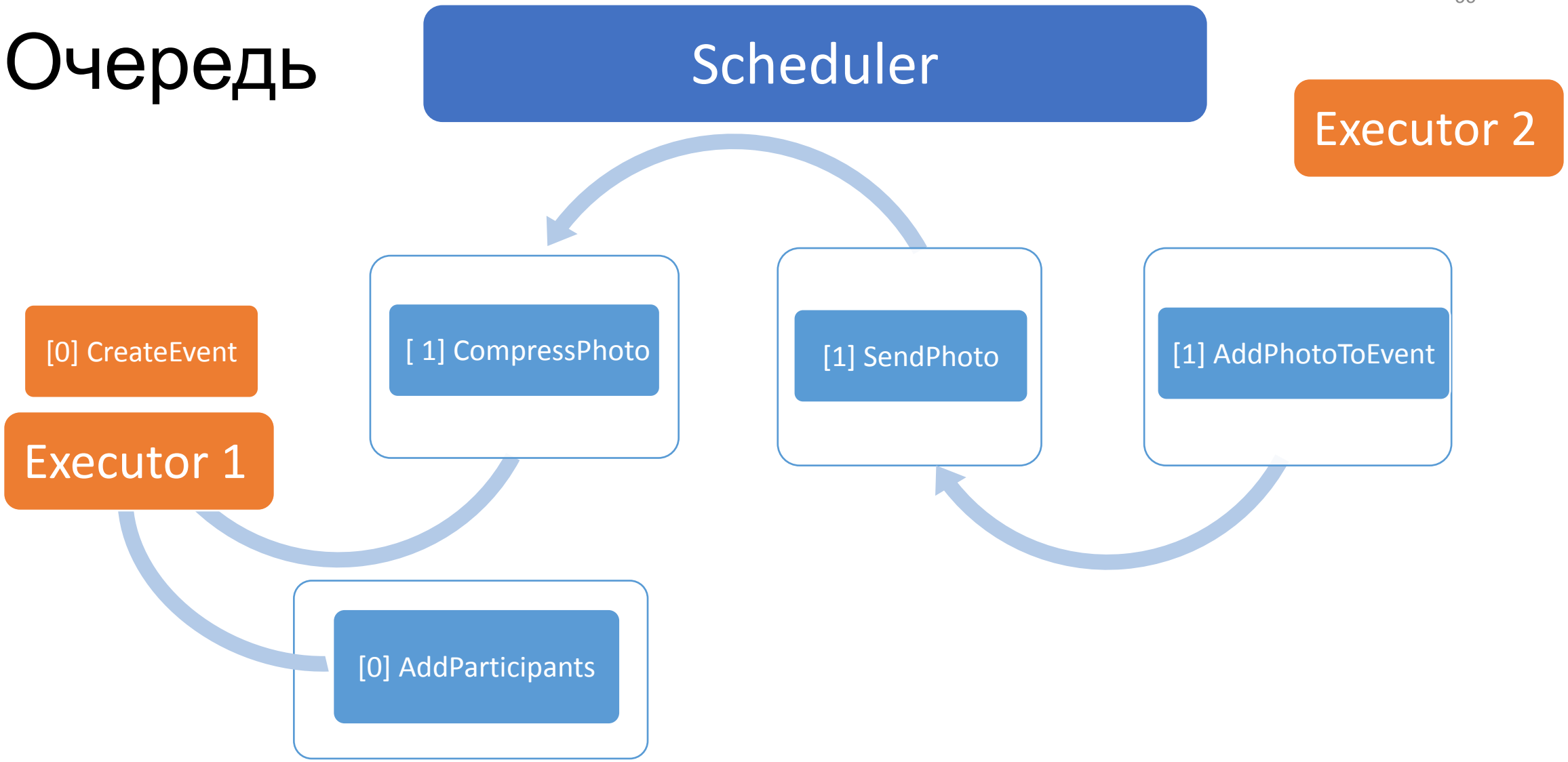
Очередь


Scheduler
Executor 1
Executor 2
CreateEvent
CompressPhoto
SendPhoto
AddPhotoToEvent
AddParticipants

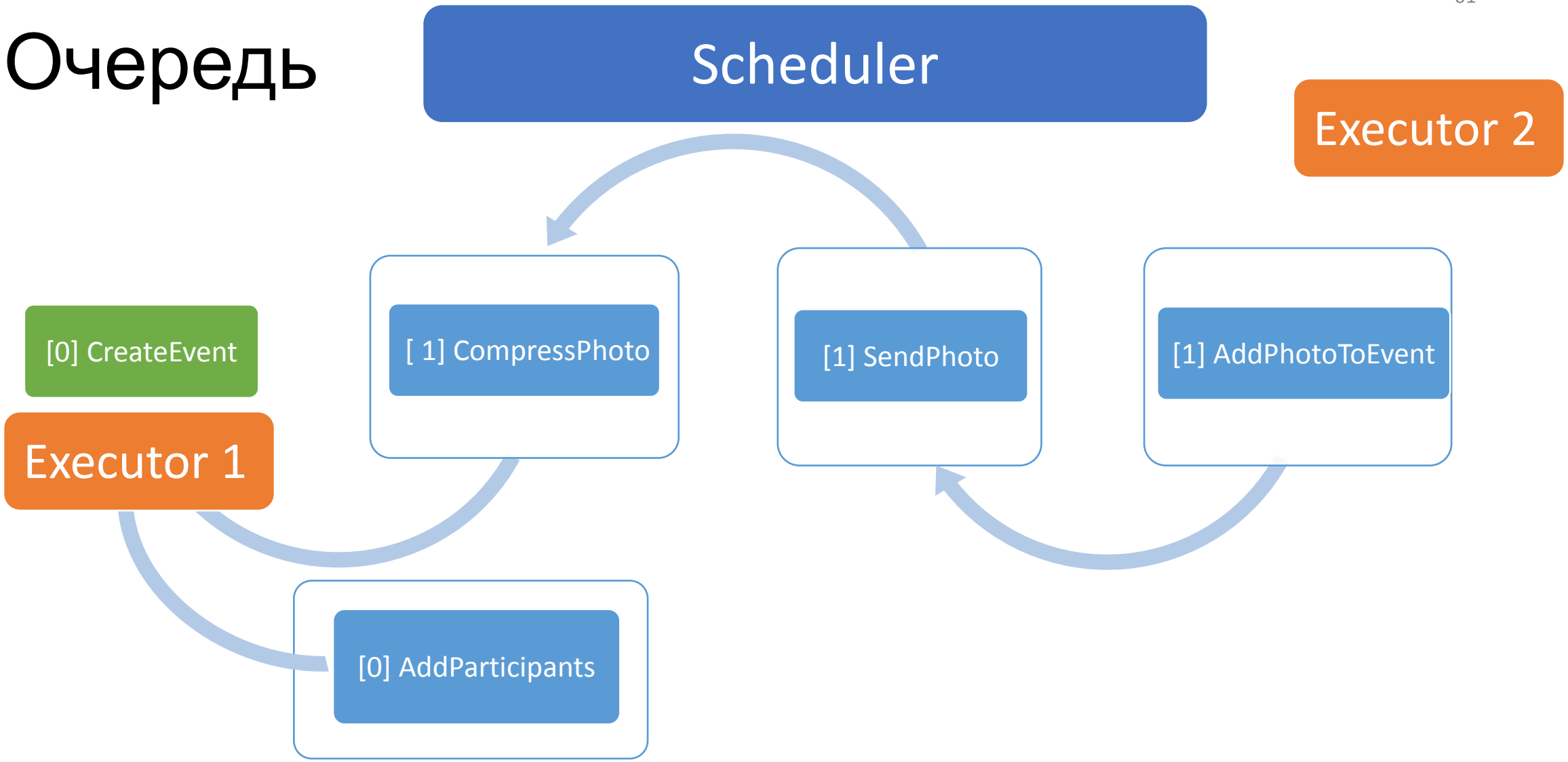
Очередь


Scheduler
Executor 1
Executor 2
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants

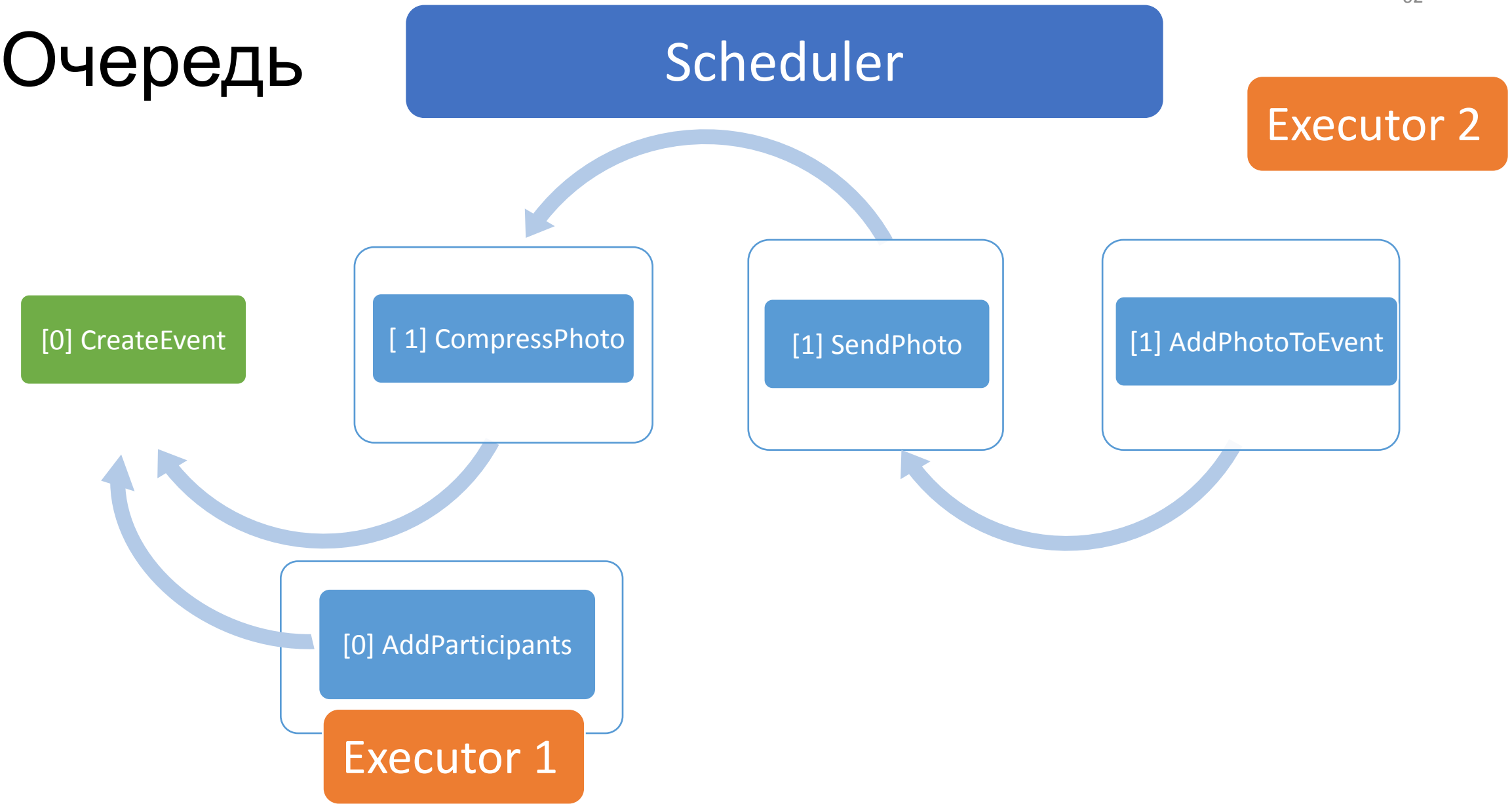
Очередь


Scheduler
Executor 2
[0] CreateEvent
Executor 1
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants

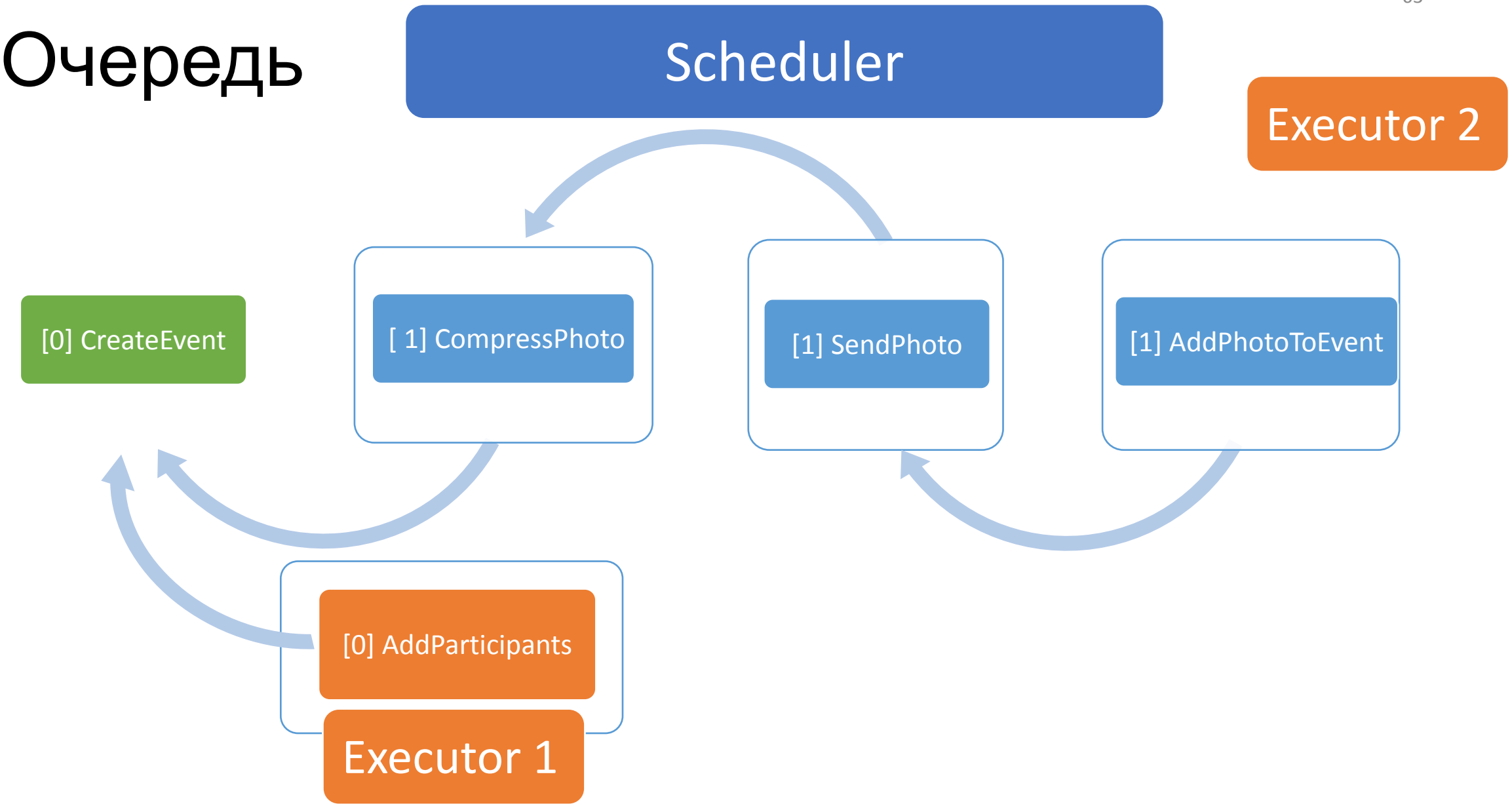
Очередь


Scheduler
Executor 2
[0] CreateEvent
Executor 1
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants

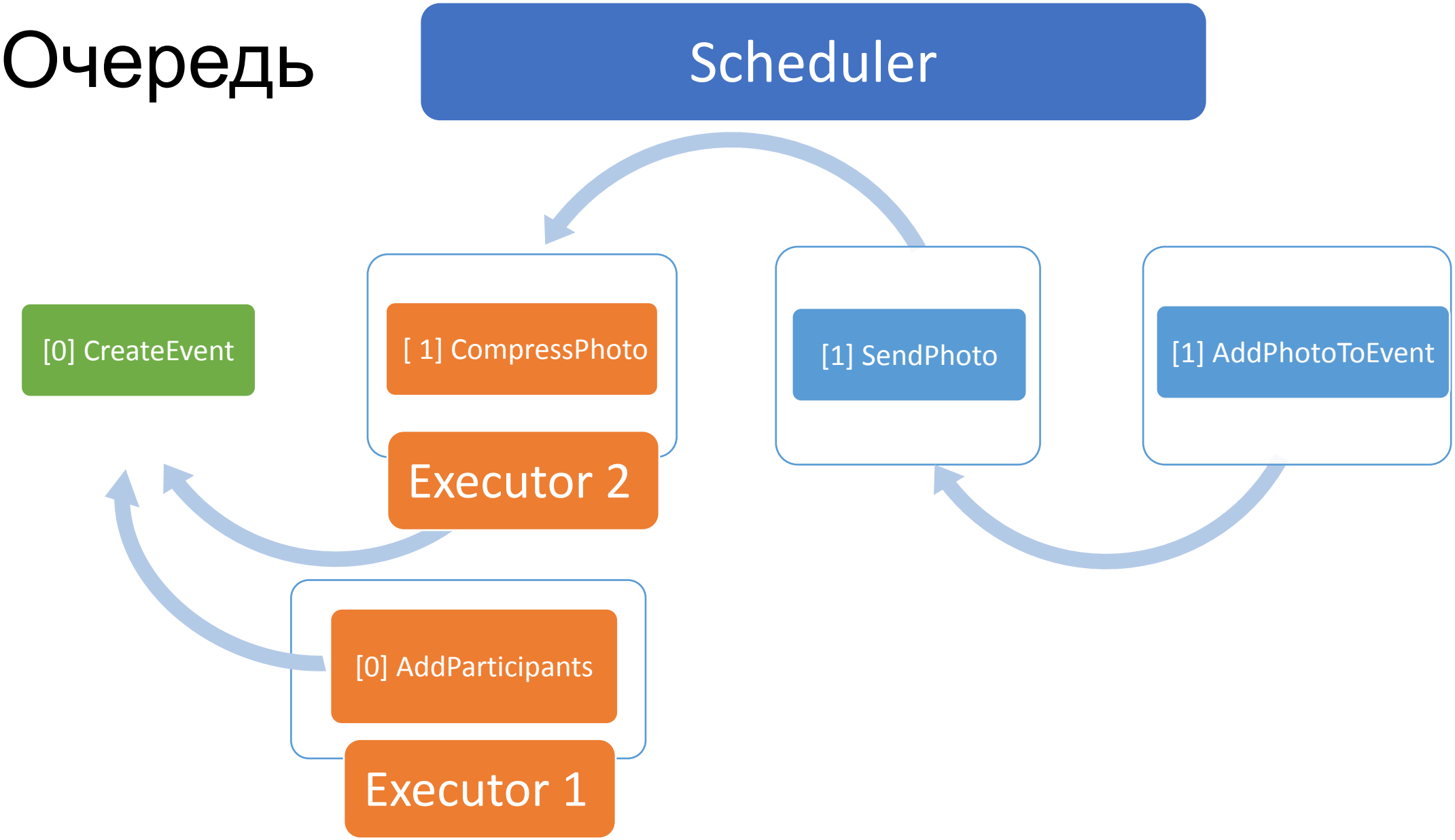
Очередь


Scheduler
Executor 2
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
Executor 1

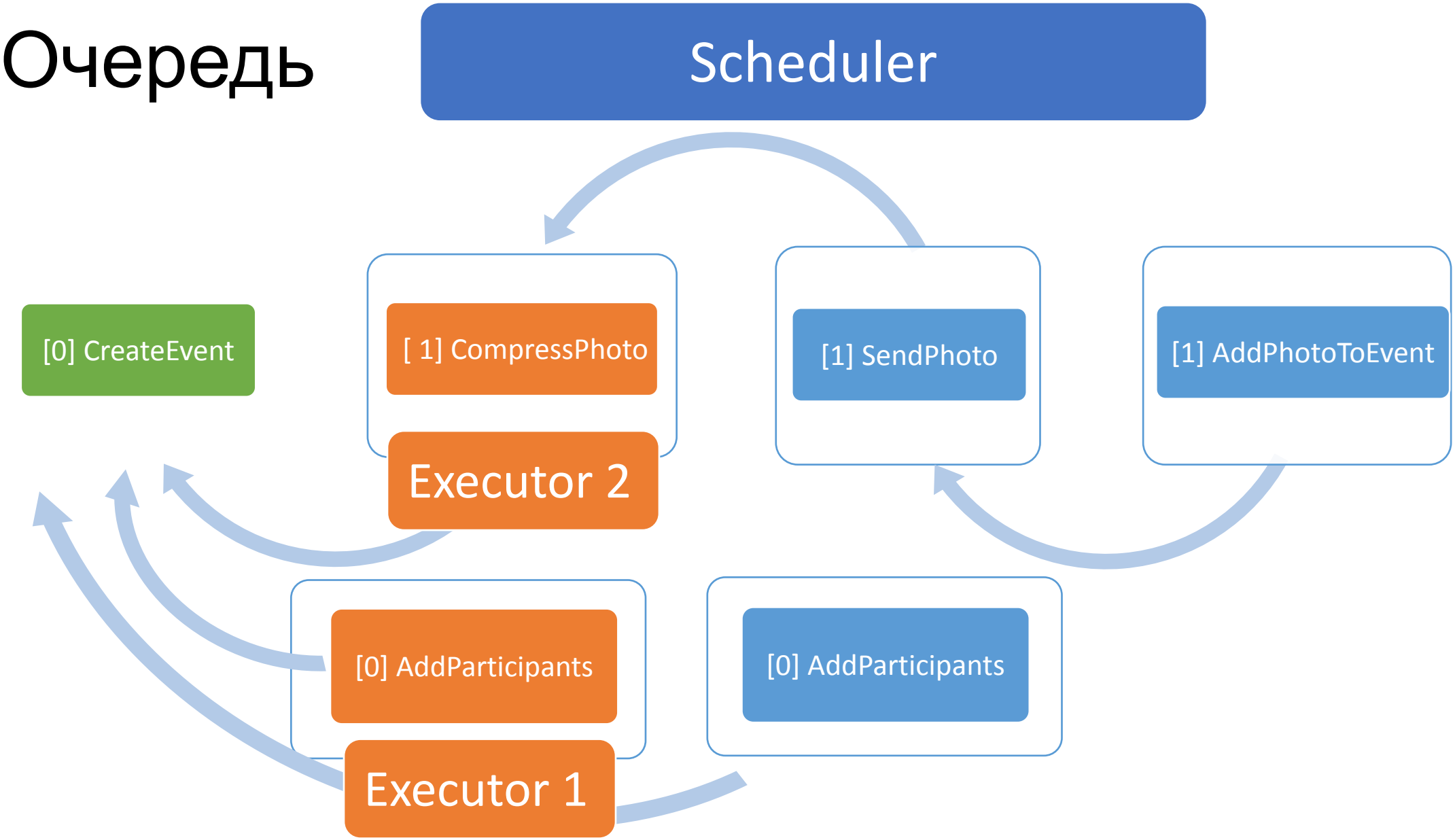
Очередь


Scheduler
Executor 2
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
Executor 1

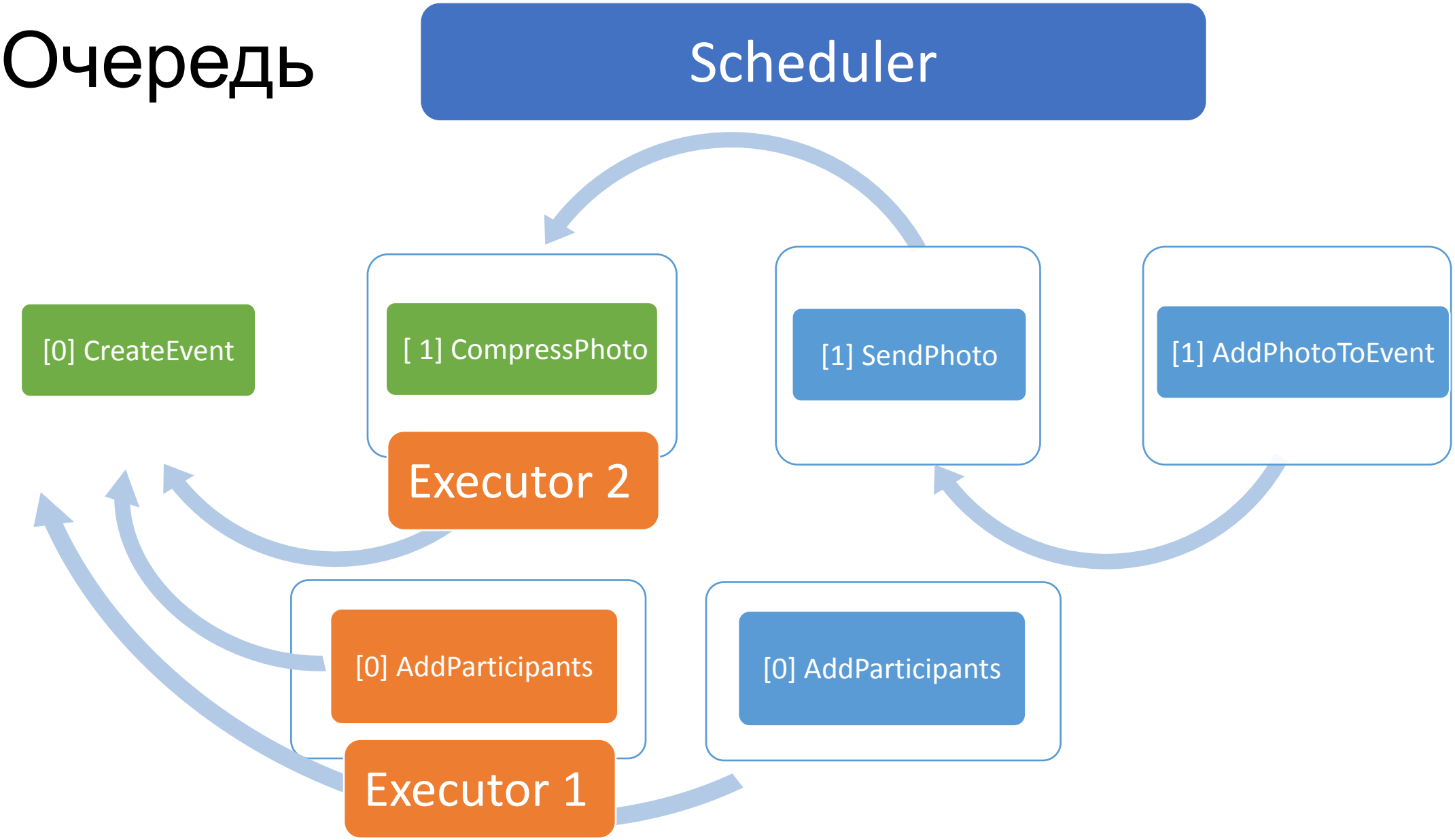
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
Executor 2
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
Executor 1

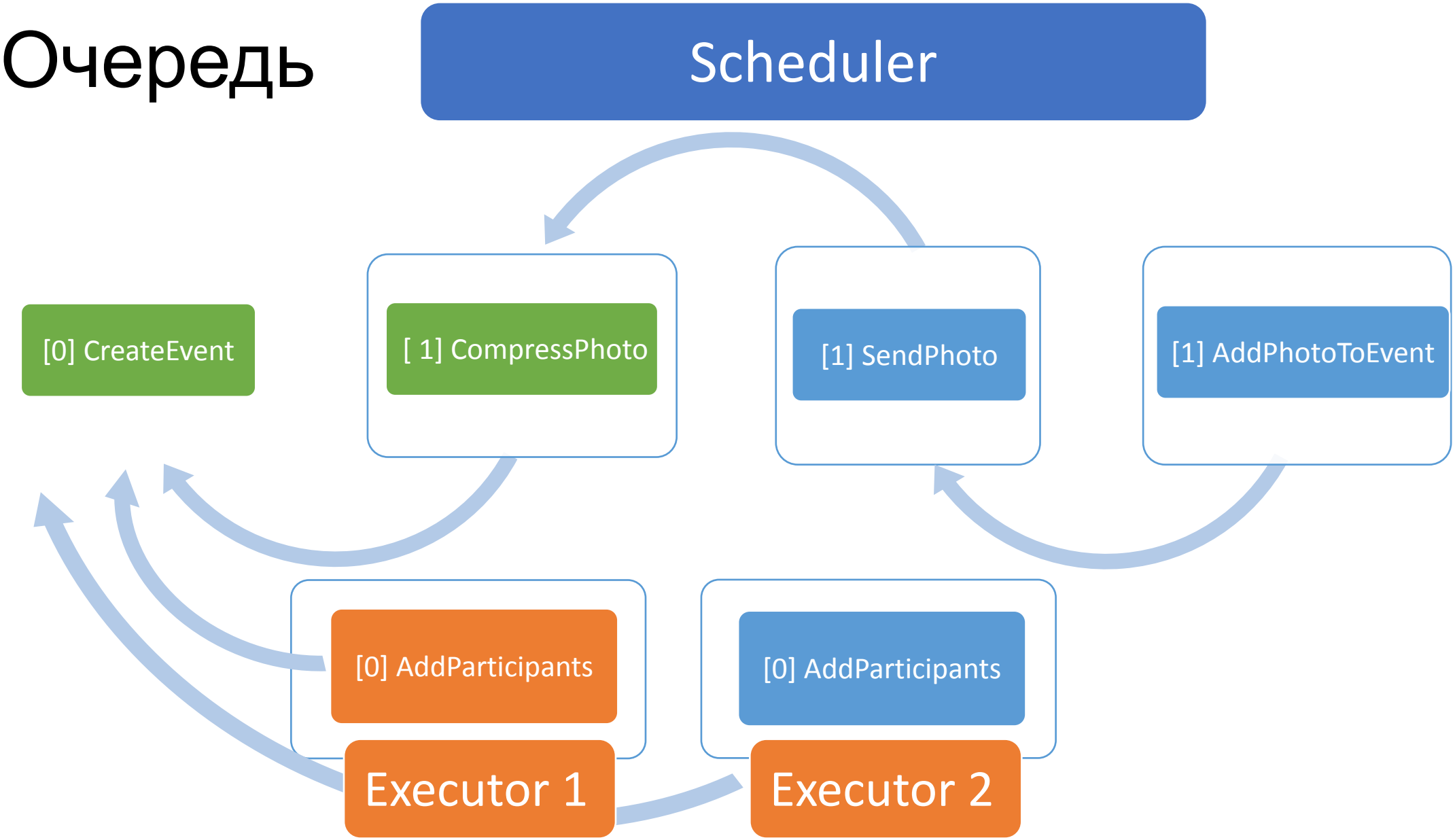
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
Executor 2
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
Executor 1
[0] AddParticipants

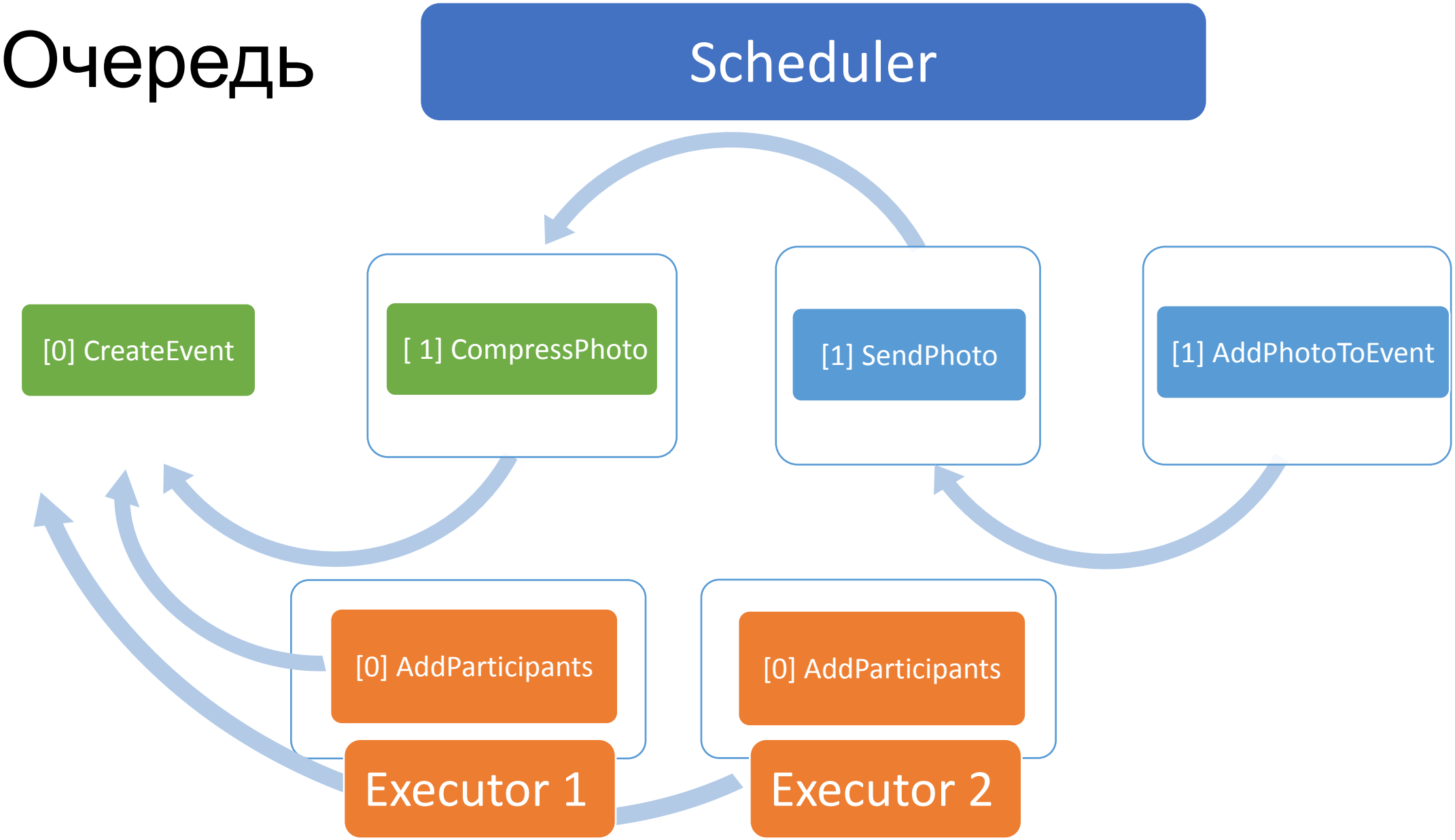
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
Executor 2
[0] AddParticipants
[0] AddParticipants
Executor 1

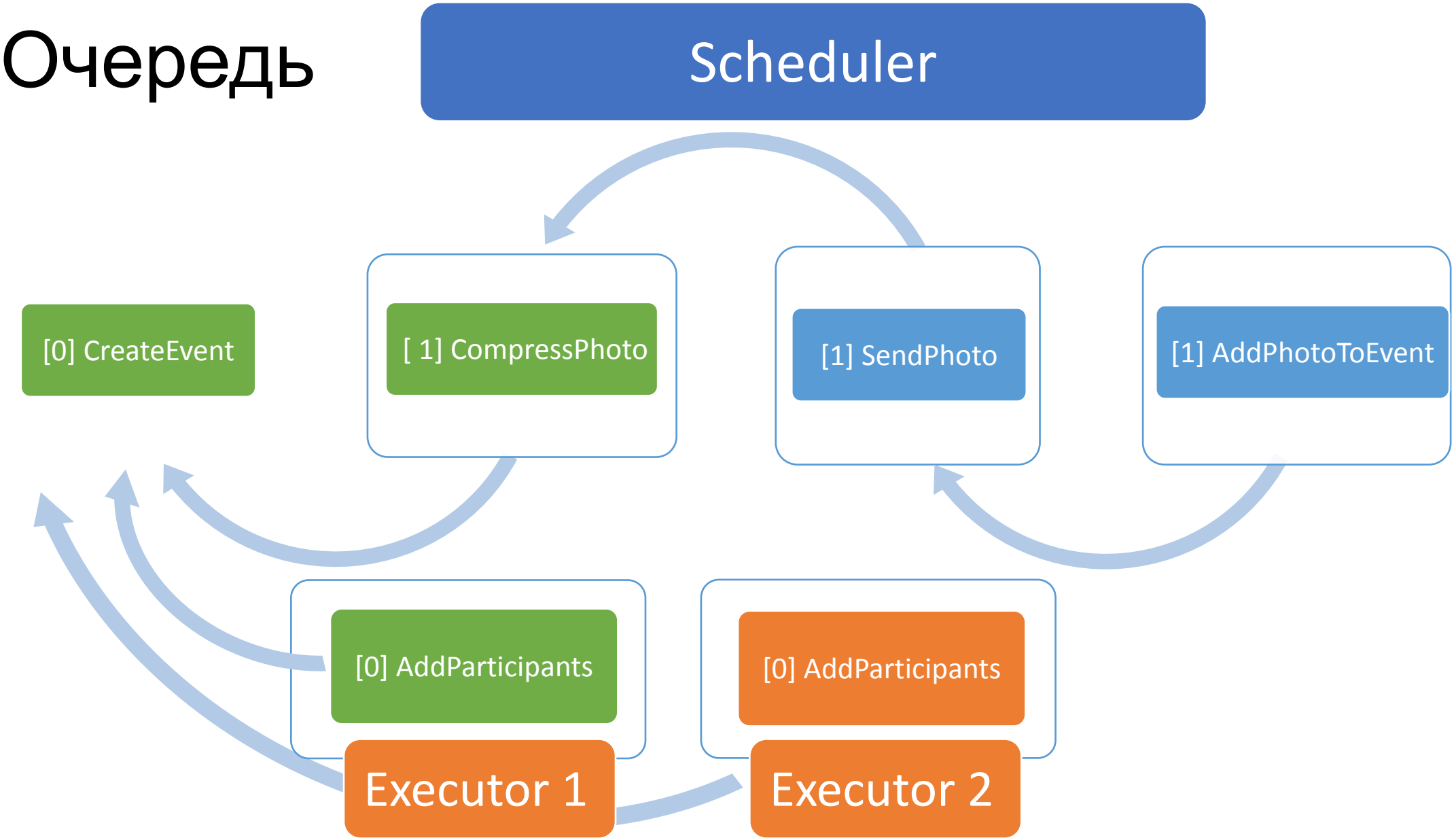
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
[0] AddParticipants
Executor 1
Executor 2

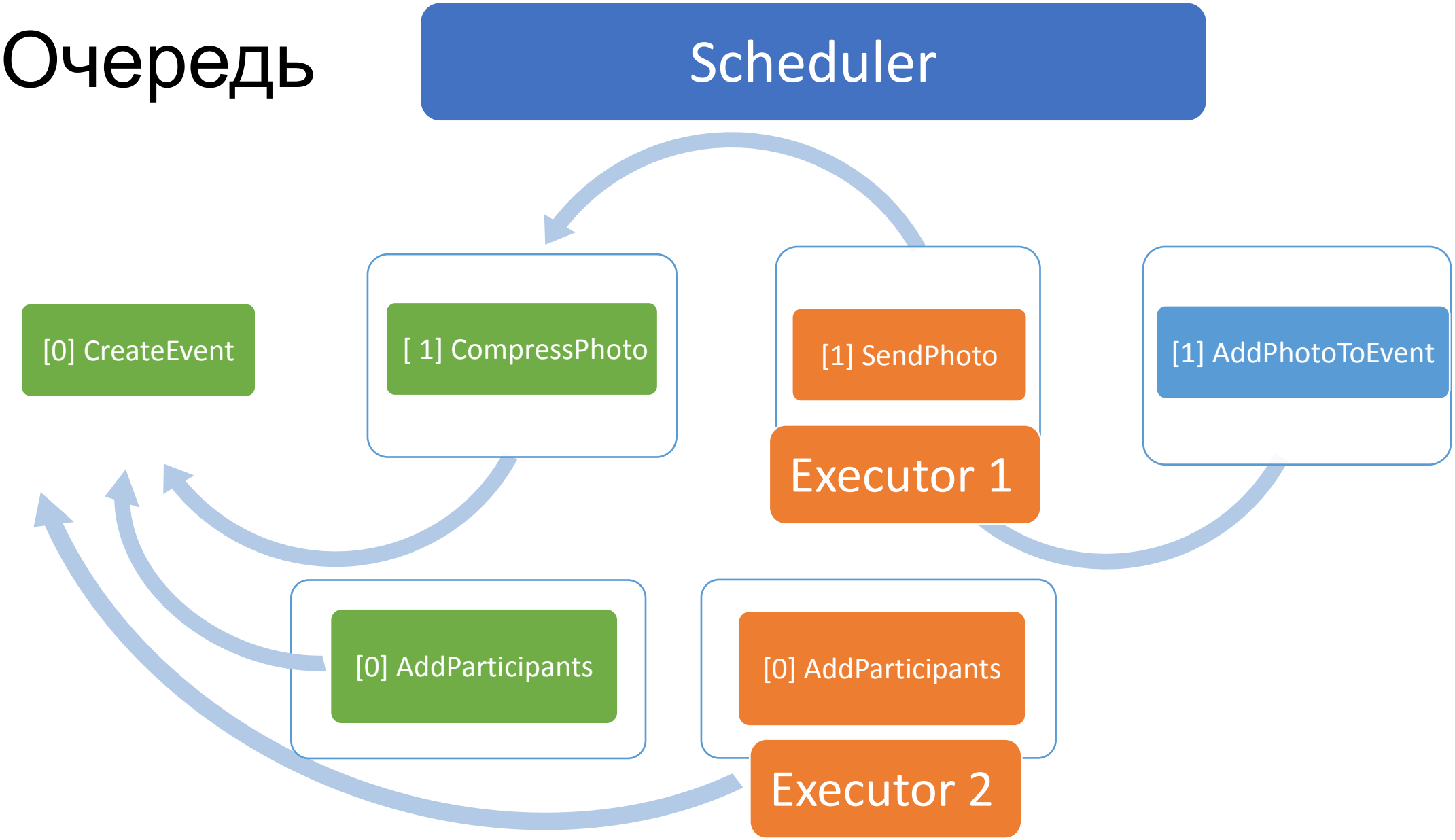
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
[0] AddParticipants
Executor 1
Executor 2

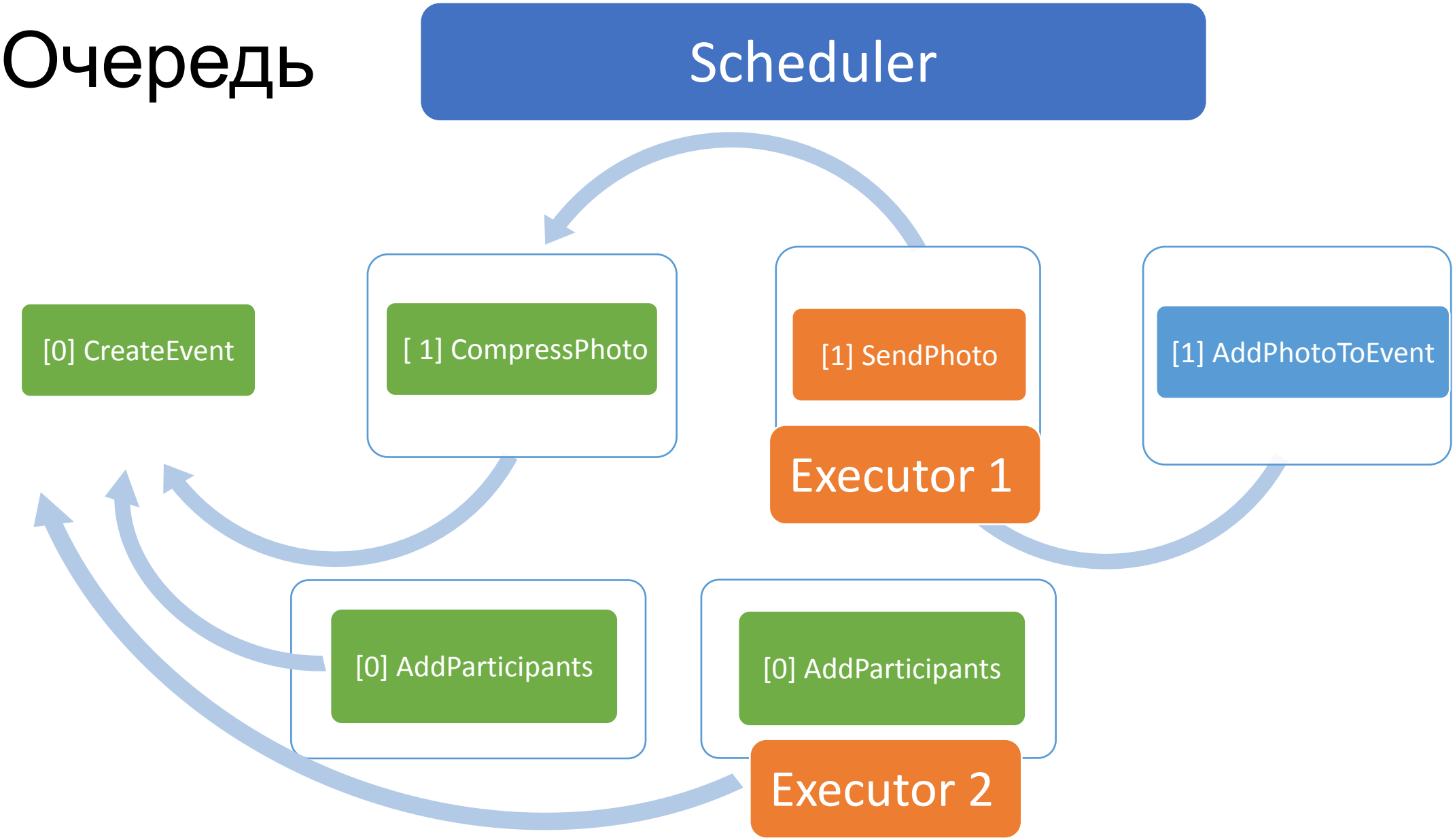
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
[0] AddParticipants
Executor 1
Executor 2

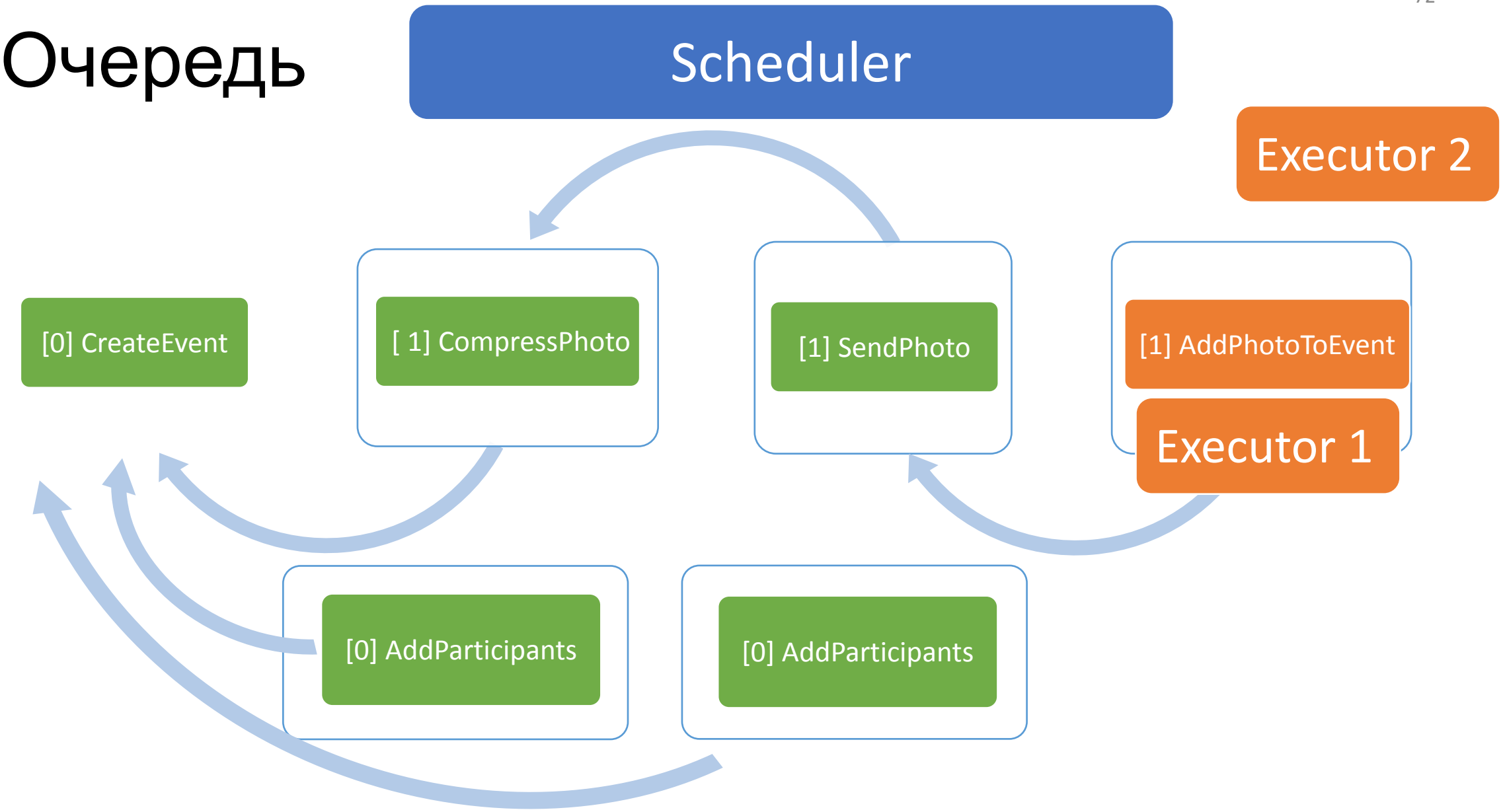
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
Executor 1
[0] AddParticipants
[0] AddParticipants
Executor 2

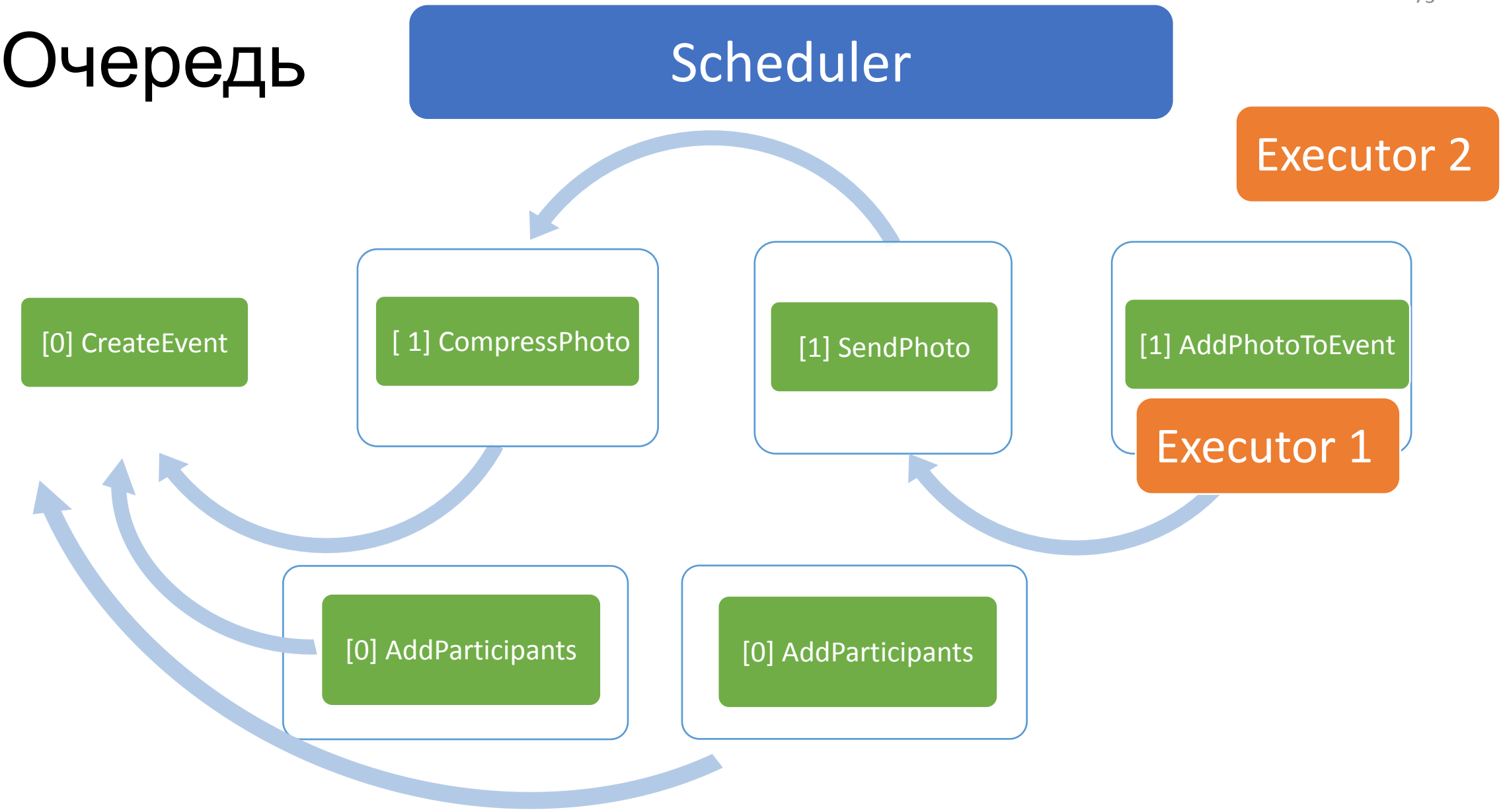
Очередь


Scheduler
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
Executor 1
[0] AddParticipants
[0] AddParticipants
Executor 2

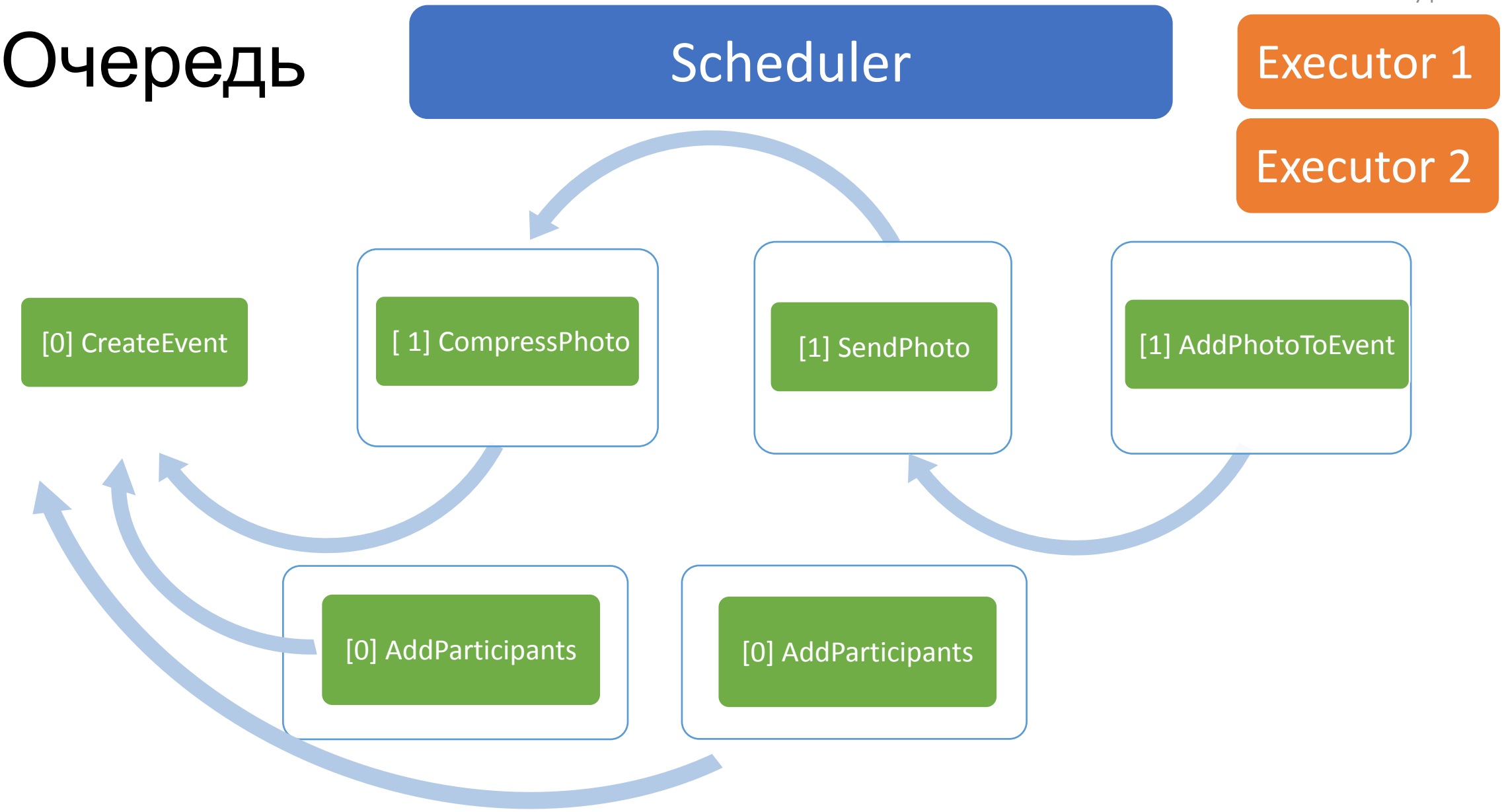
Очередь


Scheduler
Executor 2
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
Executor 1
[0] AddParticipants
[0] AddParticipants

Очередь


Scheduler
Executor 2
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
Executor 1
[0] AddParticipants
[0] AddParticipants

Очередь


Scheduler
Executor 1
Executor 2
[0] CreateEvent
[1] CompressPhoto
[1] SendPhoto
[1] AddPhotoToEvent
[0] AddParticipants
[0] AddParticipants

Преимущества

- Удобно разрабатывать
- Гарантированная доставка
- Просто визуализировать

Недостатки: сложно

Недостатки

- Необходимость синхронизировать очередь
- Сложный, витиеватый код

И ещё пара рецептов

Взаимоисключение задач

AddParticipant

Взаимоисключение задач

AddParticipant

RemoveParticipant

Взаимоисключение задач

```
public class AddParticipantRequest : Request
{
  public int EventId { get; }
  public int UserId { get; }

  public override RequestConnectedType GetConnectedType(Request
anotherRequest) =>
    anotherRequest is RemoveParticipantRequest r && r.UserId == UserId
      ? RequestConnectedType.MutuallyExclusive
      : base.GetConnectedType(anotherRequest);
}
```

Взаимоисключение задач

```
public class AddParticipantRequest : Request
{
  public int EventId { get; }
  public int UserId { get; }

  public override RequestConnectedType GetConnectedType(Request
anotherRequest) =>
    anotherRequest is RemoveParticipantRequest r && r.UserId == UserId
      ? RequestConnectedType.MutuallyExclusive
      : base.GetConnectedType(anotherRequest);
}
```

Взаимоисключение задач

```
public class AddParticipantRequest : Request
{
  public int EventId { get; }
  public int UserId { get; }

  public override RequestConnectedType GetConnectedType(Request
anotherRequest) =>
    anotherRequest is RemoveParticipantRequest r && r.UserId == UserId
      ? RequestConnectedType.MutuallyExclusive
      : base.GetConnectedType(anotherRequest);
}
```

Взаимоисключение задач

```
public class AddParticipantRequest : Request
{
  public int EventId { get; }
  public int UserId { get; }

  public override RequestConnectedType GetConnectedType(Request
anotherRequest) =>
    anotherRequest is RemoveParticipantRequest r && r.UserId == UserId
      ? RequestConnectedType.MutuallyExclusive
      : base.GetConnectedType(anotherRequest);
}
```

Взаимоисключение задач

```
public class AddParticipantRequest : Request
{
  public int EventId { get; }
  public int UserId { get; }

  public override RequestConnectedType GetConnectedType(Request
anotherRequest) =>
    anotherRequest is RemoveParticipantRequest r && r.UserId == UserId
      ? RequestConnectedType.MutuallyExclusive
      : base.GetConnectedType(anotherRequest);
}
```

Взаимоисключение задач

AddParticipant

RemoveParticipant

Взаимоисключение задач

Дедупликация данных

Создание события

Дедупликация данных

Создание события

Созда ю

Дедупликация данных

Создание события

Созда ю

Созда л

Дедупликация данных

Создание события

Ошибка сети

Созда ю

Созда л

Дедупликация данных

Создание события

Ошибка сети

Создание события

Созда ю

Созда л

Дедупликация данных

Создание события

Ошибка сети

Создание события

Созда ю

Созда л

Созда ю

Дедупликация данных

Создание события

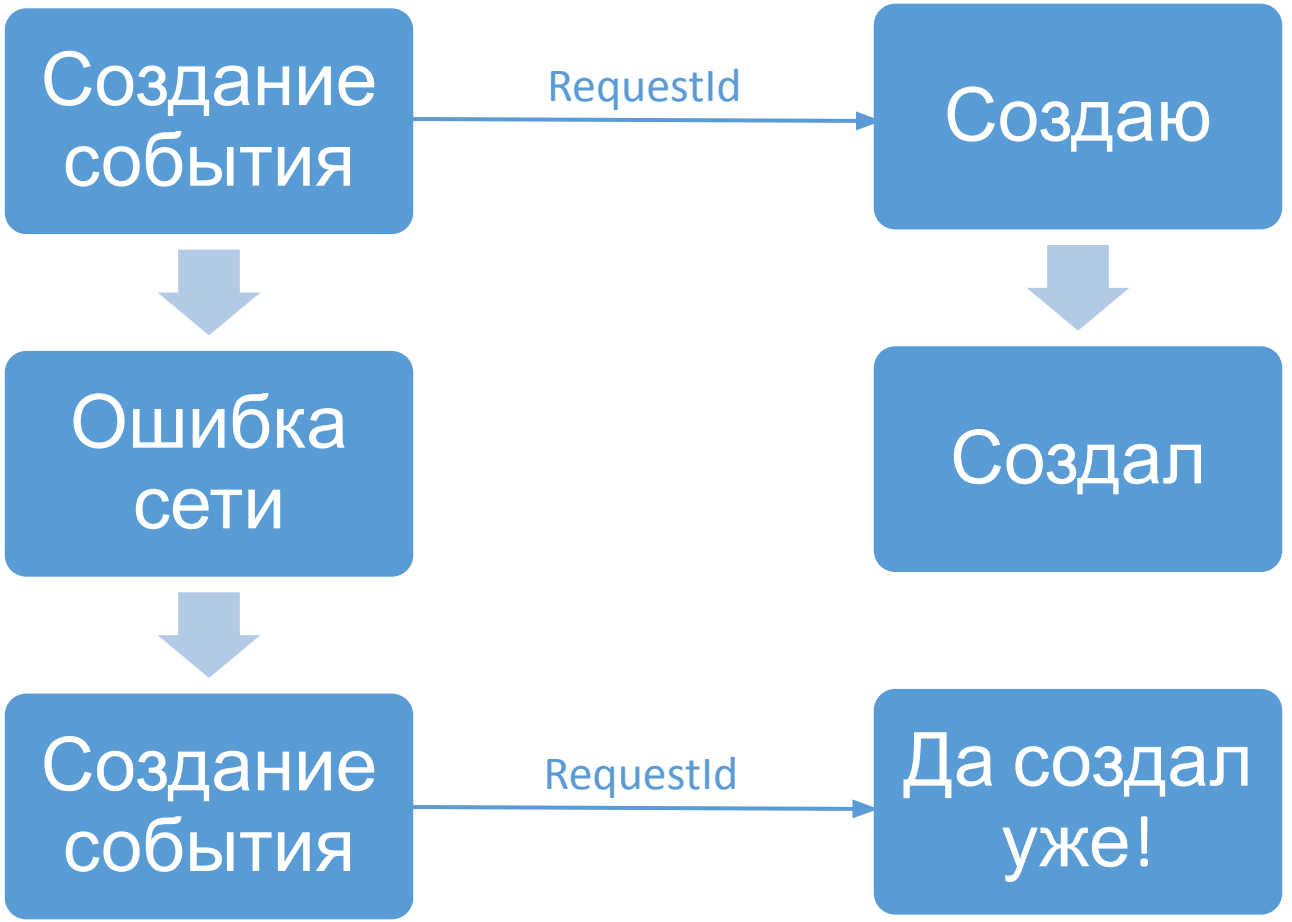
RequestId

Создаю

Ошибка сети

Создал

Дедупликация данных



Итоги

- Разбиваем задачи на атомарные подзадачи
- Сохранение и восстановление данных и состояния
- Связанные задачи выполняем последовательно
- Приоритеты задач
- Взаимоисключение задач
- Отправка RequestID на сервер и кэширование результата запроса на сервере по этому RequestID

Спасибо


Мы все доделали!
А что еще осталось?
Пофикскить баги, задеплоить, ...