INCETRO

DAO Autograph

INCETRO

Александр Лезя

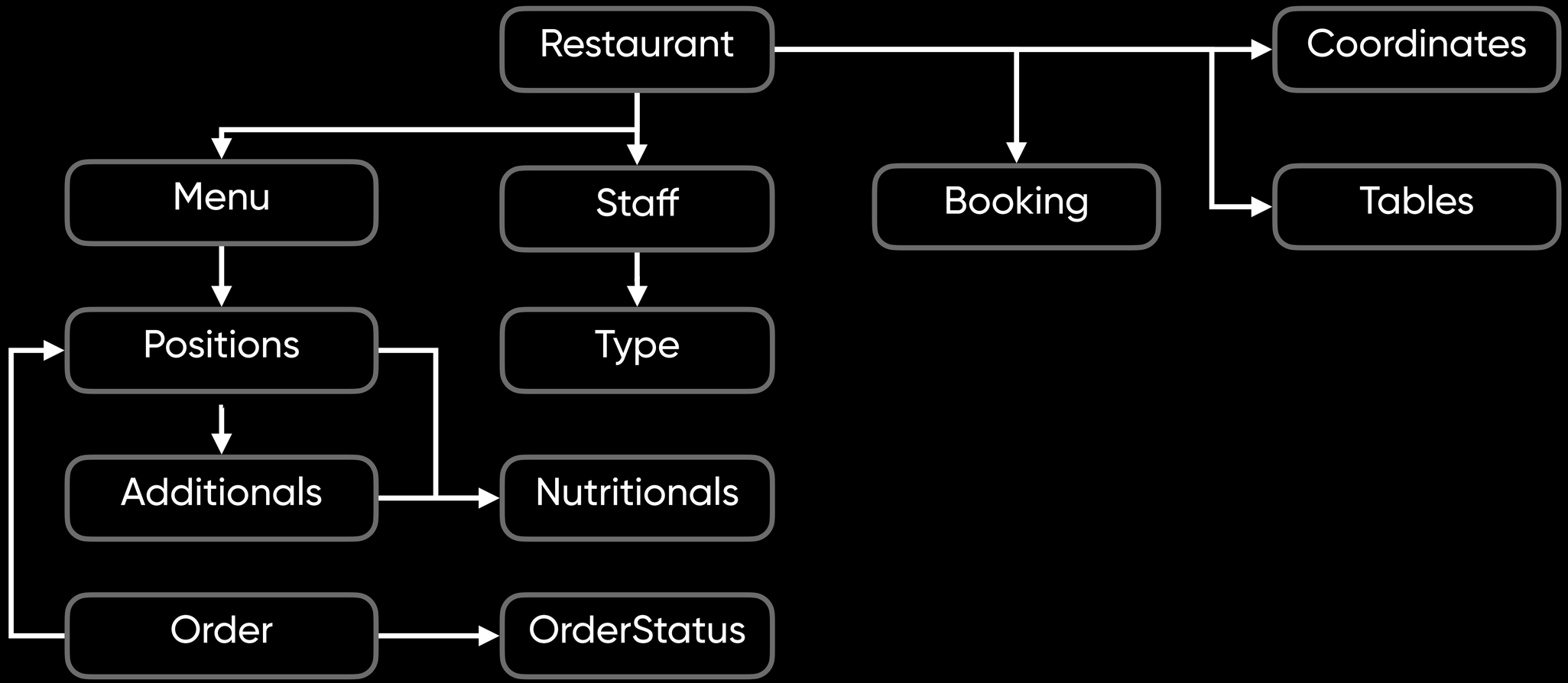
iOS Developer

INCETRO

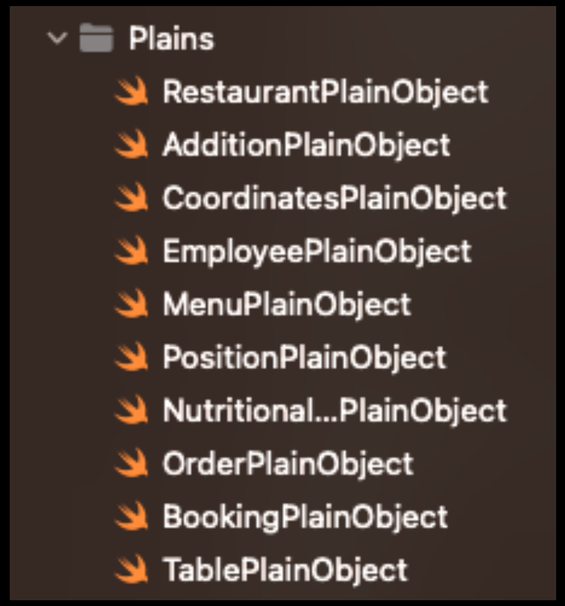
Работа DAO Autograph на примере проекта Venue

INCETRO

Схема доменной модели Venue



Plains




```swift
// MARK: - RestaurantPlainObject

/// @realm
public struct RestaurantPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Restaurant name
    public let name: String

    /// Average receipt
    public let averageReceipt: Int

    /// Restaurant menus
    public let menus: [MenuPlainObject]

    /// Restaurant staff
    public let staff: [EmployeePlainObject]

    /// Restaurant location
    public let coordinates: CoordinatesPlainObject

    /// Restaurant address
    public let address: String

    /// Restaurant bookings
    public let bookings: [BookingPlainObject]

    /// Restaurant tables
    public let tables: [TablePlainObject]
}
```

```swift
// MARK: - MenuPlainObject

/// @realm
public struct MenuPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Menu name
    public let name: String

    /// Menu positions
    public let positions: [PositionPlainObject]
}
```

```swift
// MARK: - EmployeePlainObject

/// @realm
public struct EmployeePlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Employee first name
    public let firstName: String

    /// Employee middle name
    public let middleName: String

    /// Employee last name
    public let lastName: String

    /// Employee mobile number
    public let mobile: String

    /// Employee email address
    public let email: String

    /// Employee bio information
    public let bio: String

    /// Employee registration date
    public let registeredDate: Date

    /// Employee type
    public let type: EmployeeType
}
```

```swift
// MARK: - CoordinatesPlainObject

/// @realm
public struct CoordinatesPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Latitude value
    public let latitude: Double

    /// Longitude value
    public let longitude: Double
}
```

```swift
// MARK: - BookingPlainObject

/// @realm
public struct BookingPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// True if booking is active
    public let isActive: Bool

    /// Booking date
    public let date: Date

    /// Reserver name
    public let reserverName: String

    /// Reserver mobile
    public let mobile: String

    /// Reserver email
    public let email: String?

    /// Booking table
    public let table: TablePlainObject
}
```

```swift
// MARK: - TablePlainObject

/// @realm
public struct TablePlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Table number
    public let number: Int

    /// True if table is free
    public let isFree: Bool
}
```

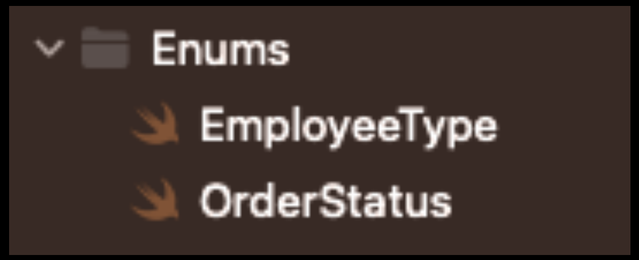
```swift
// MARK: - PositionPlainObject

/// @realm
public struct PositionPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Position name
    public let name: String

    /// Position price value
    public let price: Double

    /// Position photo url
    public let photo: URL

    /// Position nutritional values
    public let nutritionalValues: [NutritionalValuePlainObject]

    /// Position additionals
    public let additionals: [AdditionPlainObject]
}
```

INCETRO

```swift
// MARK: - AdditionPlainObject

/// @realm
public struct AdditionPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Additional name
    public let name: String

    /// Additional wight value
    public let weight: Double

    /// Additional price value
    public let price: Double

    /// Additional nutritional values
    public let nutritionalValues: [NutritionalValuePlainObject]
}
```

```swift
// MARK: - NutritionalValuePlainObject

/// @realm
public struct
NutritionalValuePlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Nutritional value name
    public let name: String

    /// Nutritional value
    public let value: Double
}
```

```swift
// MARK: - OrderPlainObject

/// @realm
public struct OrderPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Order status
    public let status: OrderStatus

    /// Order positions
    public let positions: [PositionPlainObject]

    /// Order discount value
    public let discount: Double?

    /// Order created date
    public let createdAt: Date

    /// Order total price value
    var totalPrice: Double {
        positions.map(\.price).reduce(0, +)
    }

    /// Order total price with discount value
    var totalPriceWithDiscount: Double {
        totalPrice * (discount ?? 1)
    }
}
```




```swift
// MARK: - EmployeeType

public enum EmployeeType: Int {

    // MARK: - Cases

    case admin = 0
    case vendor
    case chef
    case waiter
    case bartender
}
```

```swift
// MARK: - OrderStatus

public enum OrderStatus: Int {

    // MARK: - Cases

    case accepted = 0
    case preparing
    case served
}
```

Что дальше?

А можно как-то без этого?

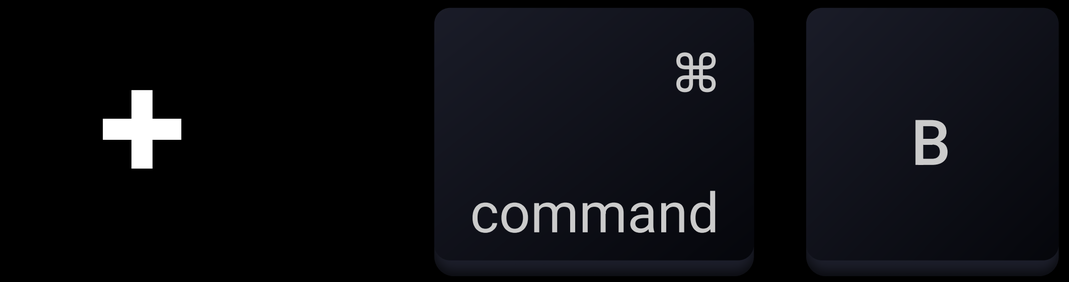
DAO Autograph

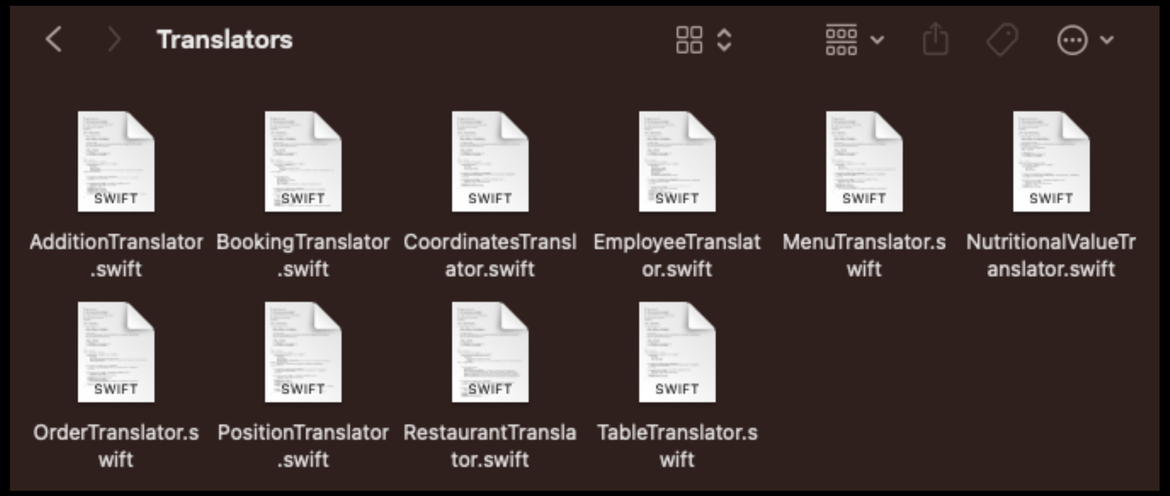
Генерация на основе моделей PlainObject

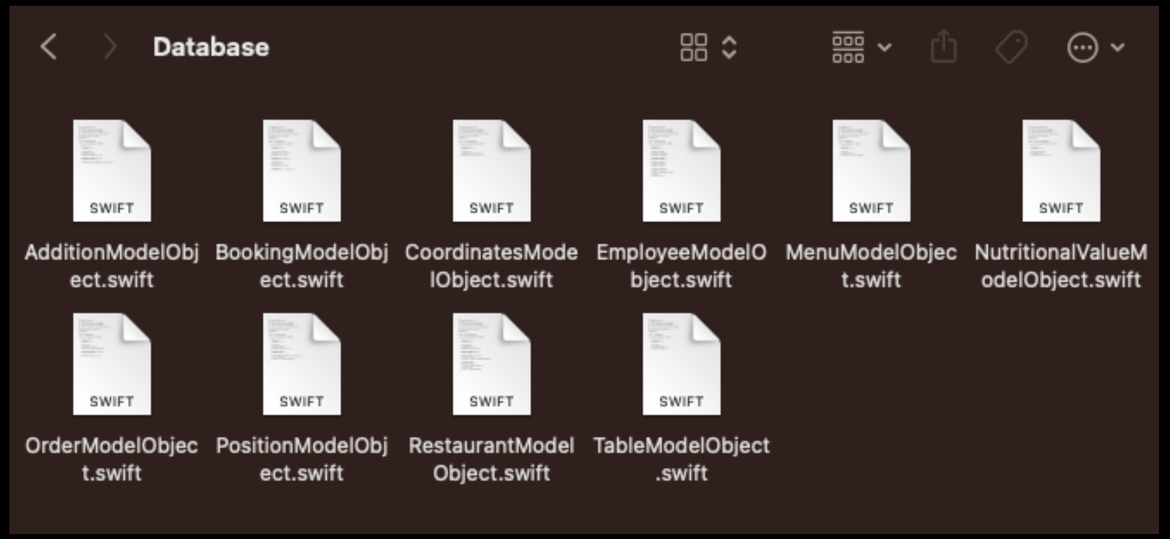
INCETRO

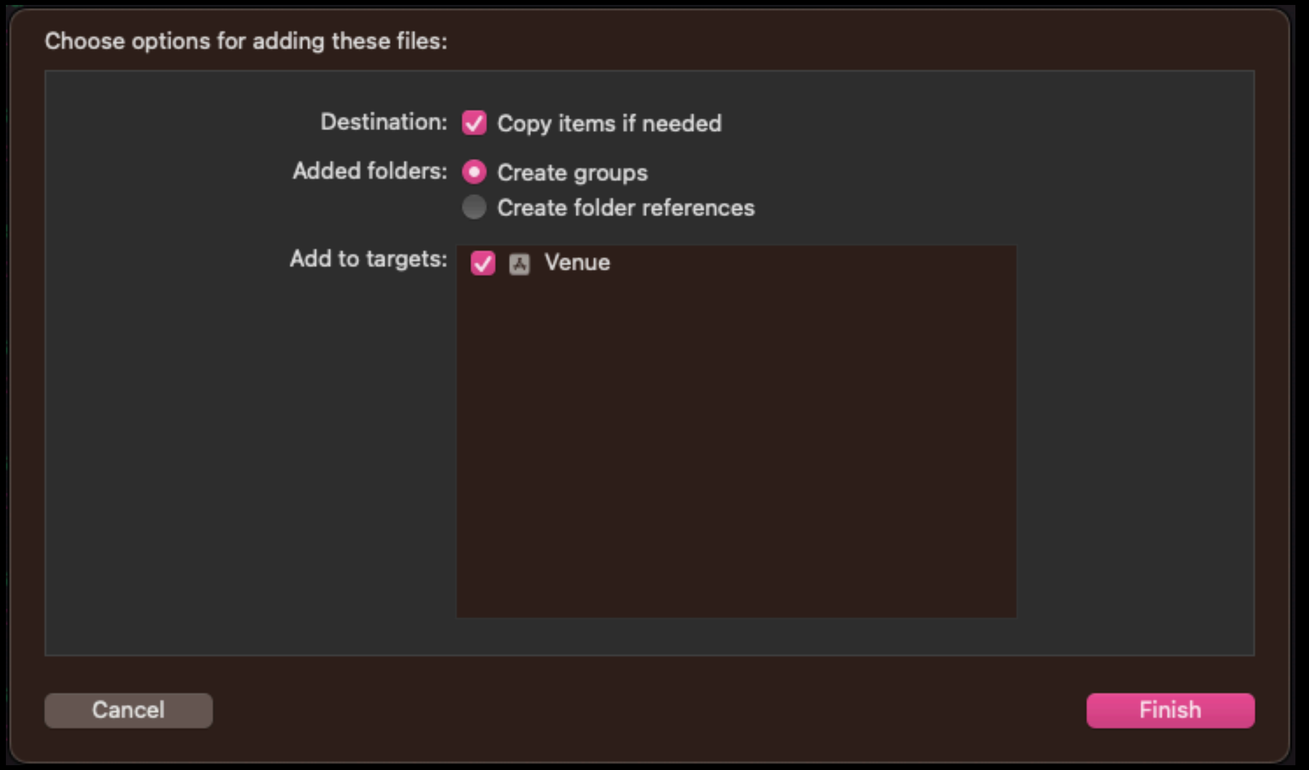
```
DAO_AUTOGRAPH_PATH=Codegen/dao-autograph

$DAO_AUTOGRAPH_PATH \
    -translators "$SRCROOT/$PROJECT_NAME/App/BusinessLayer/Translators" \
    -plains "$SRCROOT/$PROJECT_NAME/App/Models/Plains" \
    -models "$SRCROOT/$PROJECT_NAME/App/Models/Database" \
    -enums "$SRCROOT/$PROJECT_NAME/App/Models/Enums" \
    -project_name $PROJECT_NAME \
    -accessibility "public"
```

```swift
// MARK: - RestaurantPlainObject

/// @realm
public struct RestaurantPlainObject {
```




Translators
AdditionTranslator.swift
BookingTranslator.swift
CoordinatesTranslator.swift
EmployeeTranslator.swift
MenuTranslator.swift
NutritionalValueTranslator.swift
OrderTranslator.swift
PositionTranslator.swift
RestaurantTranslator.swift
TableTranslator.swift


Database
AdditionModelObject.swift
BookingModelObject.swift
CoordinatesModelObject.swift
EmployeeModelObject.swift
MenuModelObject.swift
NutritionalValueModelObject.swift
OrderModelObject.swift
PositionModelObject.swift
RestaurantModelObject.swift
TableModelObject.swift


Choose options for adding these files:
Destination:
Copy items if needed
Added folders:
Create groups
Create folder references
Add to targets:
Venue
Cancel
Finish

INCETRO

1. Добавление нового свойства в модель
2. Удаление свойства из модели

```swift
// MARK: - EmployeePlainObject

/// @realm
public struct EmployeePlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Employee first name
    public let firstName: String

    /// Employee middle name
    public let middleName: String

    /// Employee last name
    public let lastName: String

    /// Employee mobile number
    public let mobile: String

    /// Employee email address
    public let email: String

    /// Employee bio information
    public let bio: String

    /// Employee registration date
    public let registeredDate: Date

    /// Employee type
    public let type: EmployeeType


}
```

```swift
// MARK: - EmployeePlainObject

/// @realm
public struct EmployeePlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Employee first name
    public let firstName: String

    /// Employee middle name
    public let middleName: String

    /// Employee last name
    public let lastName: String

    /// Employee mobile number
    public let mobile: String

    /// Employee email address
    public let email: String

    /// Employee bio information
    public let bio: String

    /// Employee registration date
    public let registeredDate: Date

    /// Employee type
    public let type: EmployeeType

    /// Employee address
    public let address: String
}
```

```swift
// MARK: - Translator

extension EmployeeTranslator: Translator {

    func translate(model: DatabaseModel) throws -> PlainModel {
        EmployeePlainObject(
            id: model.id,
            firstName: model.firstName,
            middleName: model.middleName,
            lastName: model.lastName,
            mobile: model.mobile,
            email: model.email,
            bio: model.bio,
            registeredDate: model.registeredDate,
            type: EmployeeType(rawValue: model.type).unwrap()

        )
    }

    func translate(plain: PlainModel) throws -> DatabaseModel {
        let model = try employeeStorage.read(byPrimaryKey: plain.uniqueId.rawValue) ?? DatabaseModel()
        try translate(from: plain, to: model)
        return model
    }

    func translate(from plain: PlainModel, to databaseModel: DatabaseModel) throws {
        if databaseModel.uniqueId.isEmpty {
            databaseModel.uniqueId = plain.uniqueId.rawValue
        }
        databaseModel.id = plain.id
        databaseModel.firstName = plain.firstName
        databaseModel.middleName = plain.middleName
        databaseModel.lastName = plain.lastName
        databaseModel.mobile = plain.mobile
        databaseModel.email = plain.email
        databaseModel.bio = plain.bio
        databaseModel.registeredDate = plain.registeredDate
        databaseModel.type = plain.type.rawValue

    }
}
```

```swift
// MARK: - EmployeeModelObject

final class EmployeeModelObject: RealmModel {

    // MARK: - Properties

    /// Unique id
    @objc dynamic var id = ""

    /// Employee first name
    @objc dynamic var firstName = ""

    /// Employee middle name
    @objc dynamic var middleName = ""

    /// Employee last name
    @objc dynamic var lastName = ""

    /// Employee mobile number
    @objc dynamic var mobile = ""

    /// Employee email address
    @objc dynamic var email = ""

    /// Employee bio information
    @objc dynamic var bio = ""

    /// Employee registration date
    @objc dynamic var registeredDate = Date()

    /// Employee type
    @objc dynamic var type = 0



}
```

INCETRO

```swift
// MARK: - Translator

extension EmployeeTranslator: Translator {

    func translate(model: DatabaseModel) throws -> PlainModel {
        EmployeePlainObject(
            id: model.id,
            firstName: model.firstName,
            middleName: model.middleName,
            lastName: model.lastName,
            mobile: model.mobile,
            email: model.email,
            bio: model.bio,
            registeredDate: model.registeredDate,
            type: EmployeeType(rawValue: model.type).unwrap(),
            address: model.address
        )
    }

    func translate(plain: PlainModel) throws -> DatabaseModel {
        let model = try employeeStorage.read(byPrimaryKey: plain.uniqueId.rawValue) ?? DatabaseModel()
        try translate(from: plain, to: model)
        return model
    }

    func translate(from plain: PlainModel, to databaseModel: DatabaseModel) throws {
        if databaseModel.uniqueId.isEmpty {
            databaseModel.uniqueId = plain.uniqueId.rawValue
        }
        databaseModel.id = plain.id
        databaseModel.firstName = plain.firstName
        databaseModel.middleName = plain.middleName
        databaseModel.lastName = plain.lastName
        databaseModel.mobile = plain.mobile
        databaseModel.email = plain.email
        databaseModel.bio = plain.bio
        databaseModel.registeredDate = plain.registeredDate
        databaseModel.type = plain.type.rawValue
        databaseModel.address = plain.address
    }
}
```

```swift
// MARK: - EmployeeModelObject

final class EmployeeModelObject: RealmModel {

    // MARK: - Properties

    /// Unique id
    @objc dynamic var id = ""

    /// Employee first name
    @objc dynamic var firstName = ""

    /// Employee middle name
    @objc dynamic var middleName = ""

    /// Employee last name
    @objc dynamic var lastName = ""

    /// Employee mobile number
    @objc dynamic var mobile = ""

    /// Employee email address
    @objc dynamic var email = ""

    /// Employee bio information
    @objc dynamic var bio = ""

    /// Employee registration date
    @objc dynamic var registeredDate = Date()

    /// Employee type
    @objc dynamic var type = 0

    /// Employee address
    @objc dynamic var address = ""
}
```

INCETRO

Таблица соответствия типов

Type	Non-optional	Optional
Bool	dynamic var value = false	let value = RealmOptional<Bool> ()
Int	dynamic var value = 0	let value = RealmOptional<Int>()
Float	dynamic var value: Float = 0.0	let value = RealmOptional<Float> ()
Double	dynamic var value: Double = 0.0	let value = RealmOptional Double> ()
String	dynamic var value: String = ""	dynamic var value: String?
Data	dynamic var value: Data = Data()	dynamic var value: Data?
Date	dynamic var value: Date = Date ()	dynamic var value: Date?
Object	n/a: must be optional	dynamic var value: Class?
Array	let value = List<Class> ()	n/a: must be non-optional
Enum	dynamic var status = 0	n/a: must be non-optional

Enum

Каждый enum должны быть простым типом

```
// MARK: - OrderStatus

public enum OrderStatus: Int {

    // MARK: - Cases

    case accepted = 0
    case preparing
    case served
}
```

```swift
// MARK: - OrderPlainObject

/// @realm
public struct OrderPlainObject {

    // MARK: - Properties

    /// Unique id
    public let id: String

    /// Order status
    public let status: OrderStatus

    /// Order positions
    public let positions: [PositionPlainObject]

    /// Order discount value
    public let discount: Double?

    /// Order created date
    public let createdAt: Date

    /// Order total price value
    var totalPrice: Double {
        positions.map(\.price).reduce(0, +)
    }

    /// Order total price with discount value
    var totalPriceWithDiscount: Double {
        totalPrice * (discount ?? 1)
    }
}
```

```swift
// MARK: - OrderModelObject

final class OrderModelObject: RealmModel {

    // MARK: - Properties

    /// Unique id
    @objc dynamic var id = ""

    /// Order status
    @objc dynamic var status = 0

    /// Order positions
    let positions = List<PositionModelObject>()

    /// Order discount value
    let discount = RealmOptional<Double>()

    /// Order created date
    @objc dynamic var createdAt = Date()
}
```

```swift
// MARK: - Translator

extension OrderTranslator: Translator {

    func translate(model: DatabaseModel) throws -> PlainModel {
        OrderPlainObject(
            id: model.id,
            status: OrderStatus(rawValue: model.status).unwrap(),
            positions: try PositionTranslator(configuration: configuration).translate(models: Array(model.positions)),
            discount: model.discount.value,
            createdAt: model.createdAt
        )
    }

    func translate(plain: PlainModel) throws -> DatabaseModel {
        let model = try orderStorage.read(byPrimaryKey: plain.uniqueId.rawValue) ?? DatabaseModel()
        try translate(from: plain, to: model)
        return model
    }

    func translate(from plain: PlainModel, to databaseModel: DatabaseModel) throws {
        if databaseModel.uniqueId.isEmpty {
            databaseModel.uniqueId = plain.uniqueId.rawValue
        }
        databaseModel.id = plain.id
        databaseModel.status = plain.status.rawValue
        databaseModel.positions.removeAll()
        databaseModel.positions.append(objectsIn:
            try PositionTranslator(configuration: configuration).translate(plains: plain.positions)
        )
        databaseModel.discount.value = plain.discount
        databaseModel.createdAt = plain.createdAt
    }
}
```

INCETRO

Генерация сборщиков DAO и Translators

- DAOAssembly
- TranslatorsAssembly

DAOAssembly

```
// MARK: - DAOAssembly

final class DAOAssembly: CollectableAssembly {

    required init() {
    }

    func assemble(inContainer container: Container) {

        container.register(AdditionDAO.self) { resolver in
            let translator = resolver.resolve(AdditionTranslator.self).unwrap()
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return AdditionDAO(
                storage: RealmStorage<AdditionModelObject>(configuration: configuration),
                translator: translator
            )
        }

        container.register(BookingDAO.self) { resolver in
            let translator = resolver.resolve(BookingTranslator.self).unwrap()
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return BookingDAO(
                storage: RealmStorage<BookingModelObject>(configuration: configuration),
                translator: translator
            )
        }

        container.register(CoordinatesDAO.self) { resolver in
            let translator = resolver.resolve(CoordinatesTranslator.self).unwrap()
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return CoordinatesDAO(
                storage: RealmStorage<CoordinatesModelObject>(configuration: configuration),
                translator: translator
            )
        }

        container.register(EmployeeDAO.self) { resolver in
            let translator = resolver.resolve(EmployeeTranslator.self).unwrap()
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return EmployeeDAO(
                storage: RealmStorage<EmployeeModelObject>(configuration: configuration),
                translator: translator
            )
        }
```

DAOAutographAliases

```
// MARK: - Aliases

/// DAO alias for AdditionPlainObject entity
typealias AdditionDAO = DAO<RealmStorage<AdditionModelObject>, AdditionTranslator>

/// DAO alias for BookingPlainObject entity
typealias BookingDAO = DAO<RealmStorage<BookingModelObject>, BookingTranslator>

/// DAO alias for CoordinatesPlainObject entity
typealias CoordinatesDAO = DAO<RealmStorage<CoordinatesModelObject>, CoordinatesTranslator>

/// DAO alias for EmployeePlainObject entity
typealias EmployeeDAO = DAO<RealmStorage<EmployeeModelObject>, EmployeeTranslator>

/// DAO alias for MenuPlainObject entity
typealias MenuDAO = DAO<RealmStorage<MenuModelObject>, MenuTranslator>

/// DAO alias for NutritionalValuePlainObject entity
typealias NutritionalValueDAO = DAO<RealmStorage<NutritionalValueModelObject>, NutritionalValueTranslator>

/// DAO alias for OrderPlainObject entity
typealias OrderDAO = DAO<RealmStorage<OrderModelObject>, OrderTranslator>

/// DAO alias for PositionPlainObject entity
typealias PositionDAO = DAO<RealmStorage<PositionModelObject>, PositionTranslator>

/// DAO alias for RestaurantPlainObject entity
typealias RestaurantDAO = DAO<RealmStorage<RestaurantModelObject>, RestaurantTranslator>

/// DAO alias for TablePlainObject entity
typealias TableDAO = DAO<RealmStorage<TableModelObject>, TableTranslator>
```

TranslatorsAssembly

```swift
// MARK: - TranslatorsAssembly

final class TranslatorsAssembly: CollectableAssembly {

    required init() {
    }

    func assemble(inContainer container: Container) {

        container.register(AdditionTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return AdditionTranslator(configuration: configuration)
        }

        container.register(BookingTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return BookingTranslator(configuration: configuration)
        }

        container.register(CoordinatesTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return CoordinatesTranslator(configuration: configuration)
        }

        container.register(EmployeeTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return EmployeeTranslator(configuration: configuration)
        }

        container.register(MenuTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return MenuTranslator(configuration: configuration)
        }

        container.register(NutritionalValueTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return NutritionalValueTranslator(configuration: configuration)
        }

        container.register(OrderTranslator.self) { resolver in
            let configuration = resolver.resolve(RealmConfiguration.self).unwrap()
            return OrderTranslator(configuration: configuration)
        }
```

Преимущества

- Экономия времени на написании шаблонного кода
- Исключение фактора человеческой ошибки
- Быстрое редактирование БД (добавление, удаление)
- Единый источник правды

Реальный случай

RuvPro

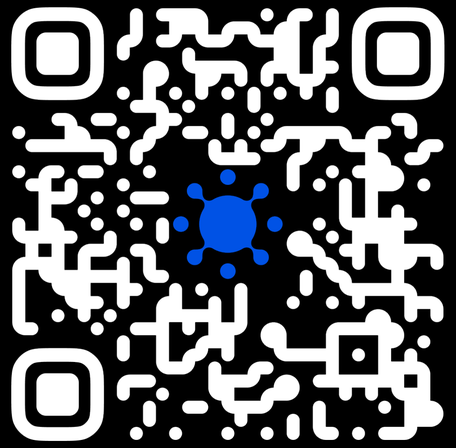
17 - PlainObject, 5 - Enum

- 34 файла (Translator & ModelObject)
- Проставление зависимостей
- Около 2000 строк кода

Как это работает?

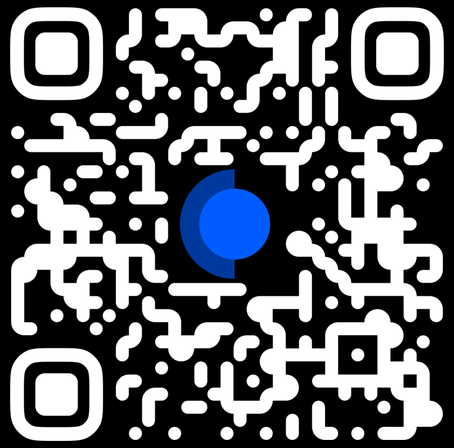
INCETRO

Ссылки на github.com



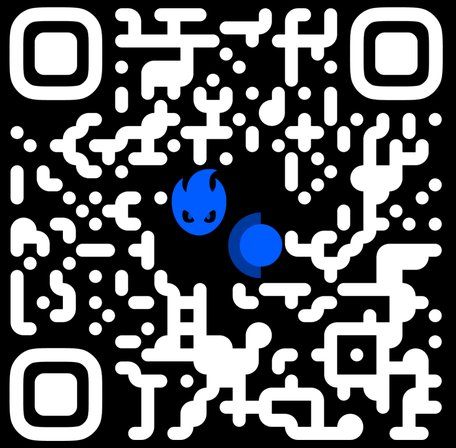
Synopsis

/Incetro/Monreau



Autograph

/Incetro/DAO



dao-autograph

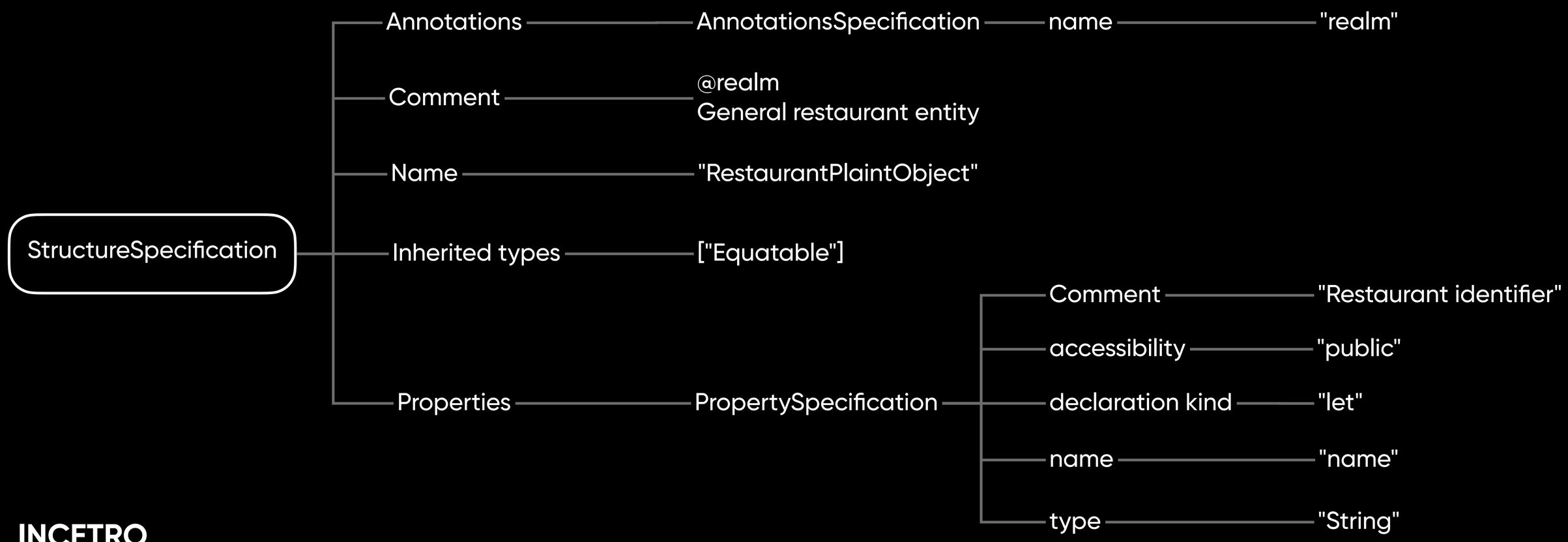
/Incetro/DAO

INCETRO

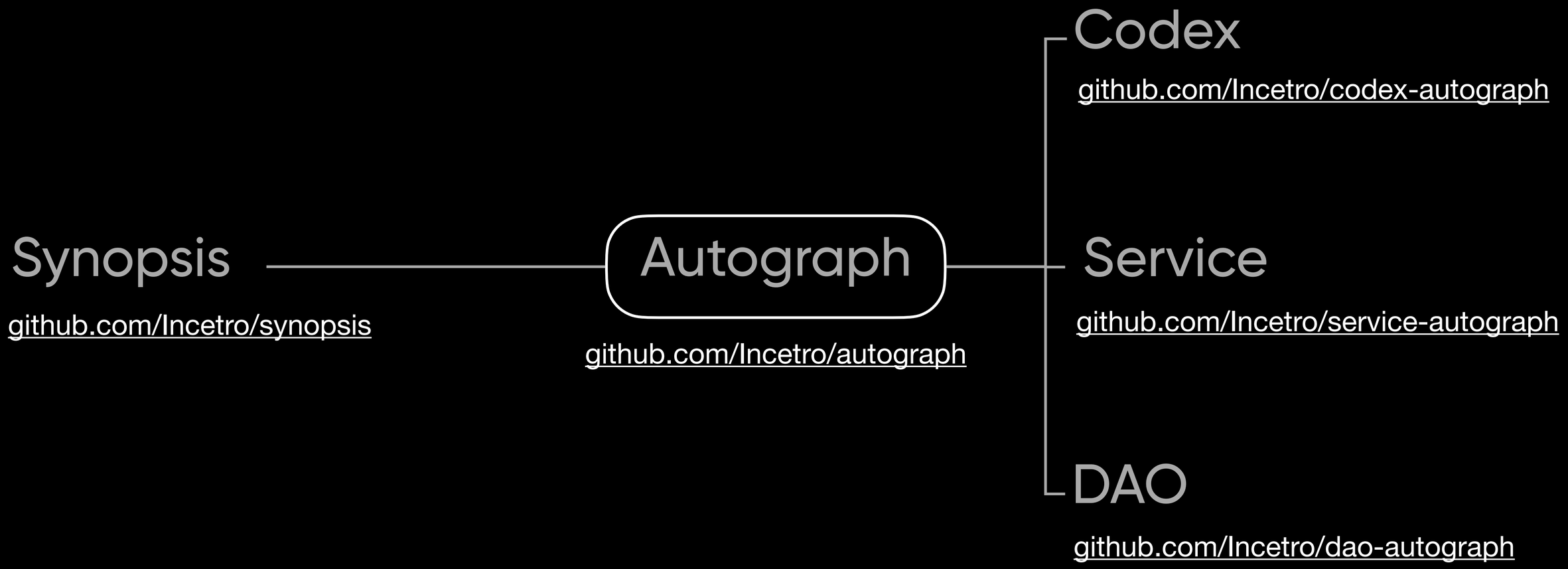
- Synopsis - сбор информации
- Autograph - ядро генерации
- Реализация - шаблоны генерации

```swift
/// @realm
/// General restaurant entity
public struct RestaurantPlainObject: Equatable {

    /// Restaurant name
    public let name: String
}
```

Парсинг Synopsis



Autograph



INCETRO

Спасибо за внимание!

INCETRO

INCETRO