

WEB3 전자지갑 풀스택 개발자

# Sony Store

- 관리자 기능 -

---

박재한 박진수 이승현

---

# 목차

CONTENT

- 1 화면 구현
- 2 데이터베이스
- 3 테스트
- 4 구현 기능

## 과제 개요



SONY

## 소니 스토어 관리자 페이지 Dashboard

스토어의 관리자들이 매출 데이터와 사용자 활동을 한눈에 파악할 수 있는 Dashboard 기능을 구현한다.

React로 프론트엔드, SpringBoot로 백엔드를 구축해 쇼핑몰의 관리자 대시보드 페이지를 개발한다.

매일 새벽 스케줄러를 통해 데이터가 집계하고, 관리자에게 쇼핑몰의 핵심 지표를 시각적으로 제공한다.

- 1) 총 매출 그래프 (1개월 간의 주별/1주일 간의 일별)
- 2) 신규 회원 추이 (1개월 간의 주별/1주일 간의 일별)
- 3) 인기 상품 순위 차트 (1개월 별/1주일 별)

PART 1

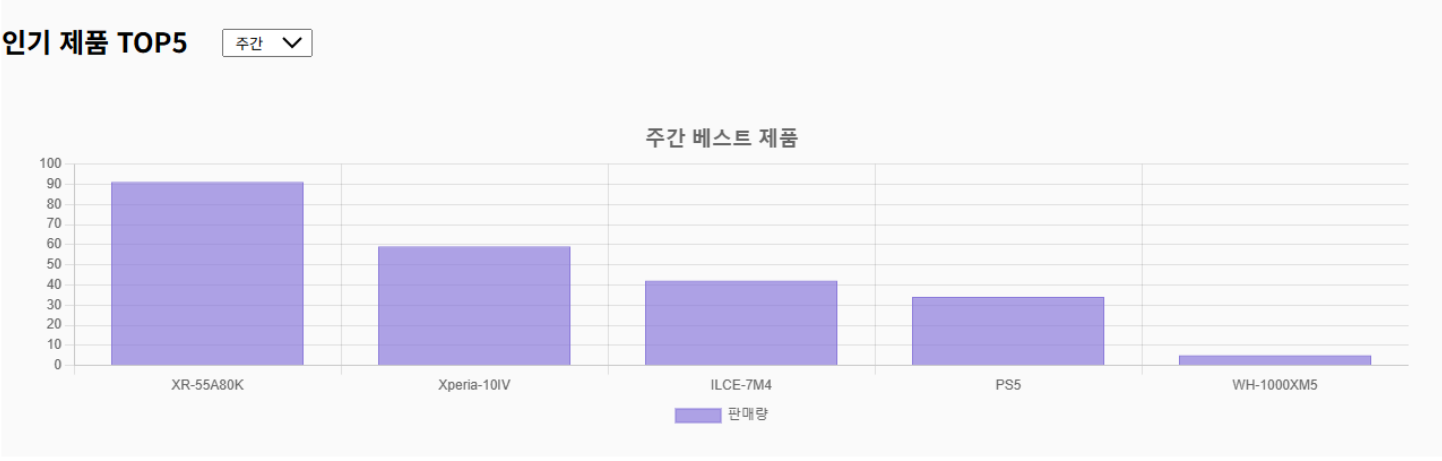
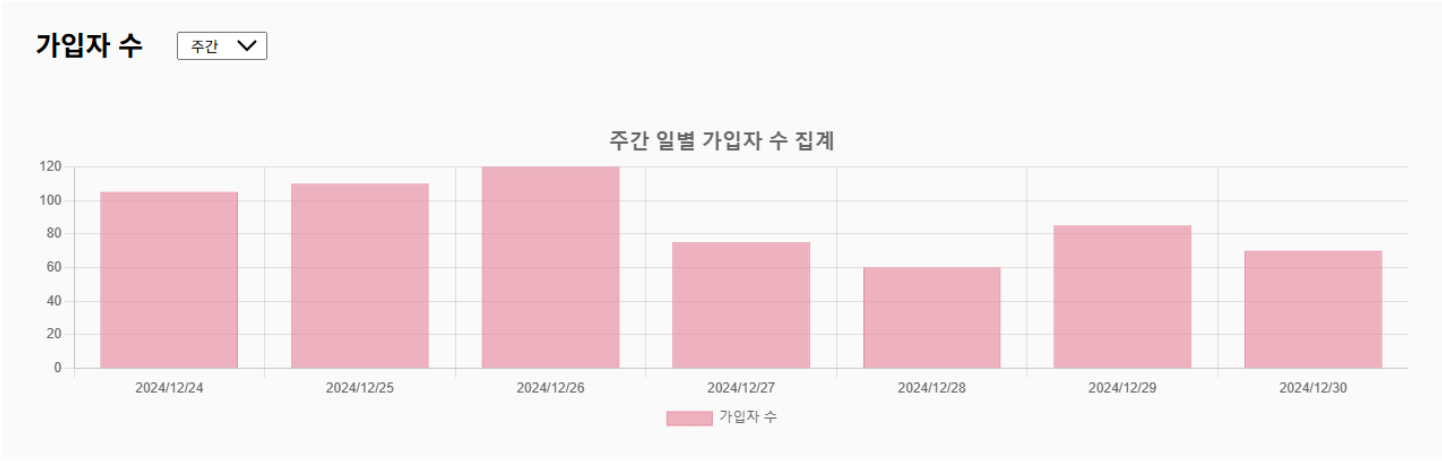
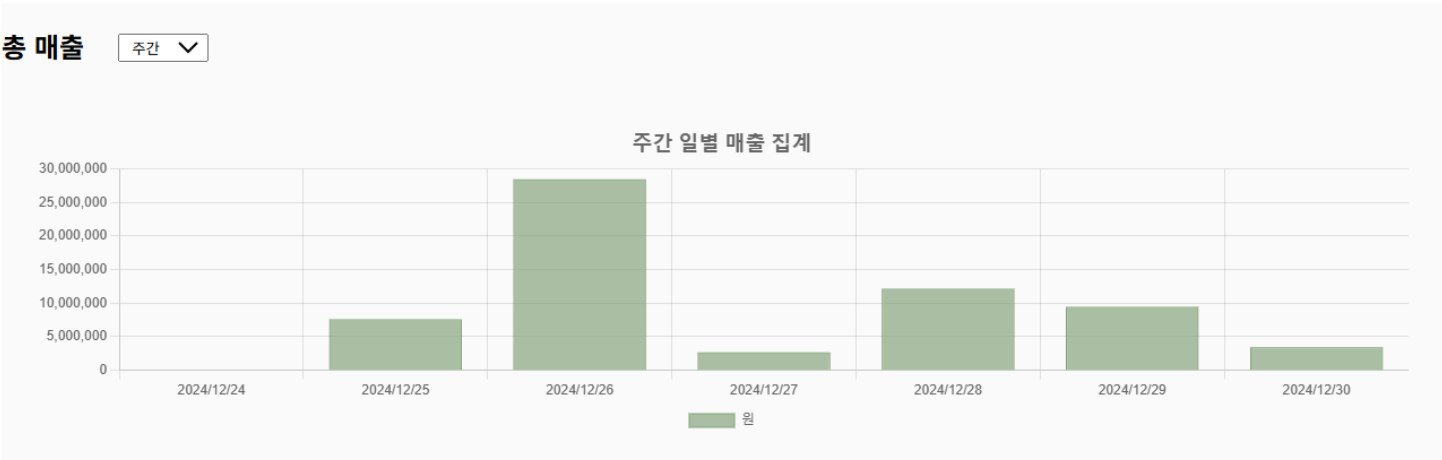
# 화면 구현

1

화면 구현

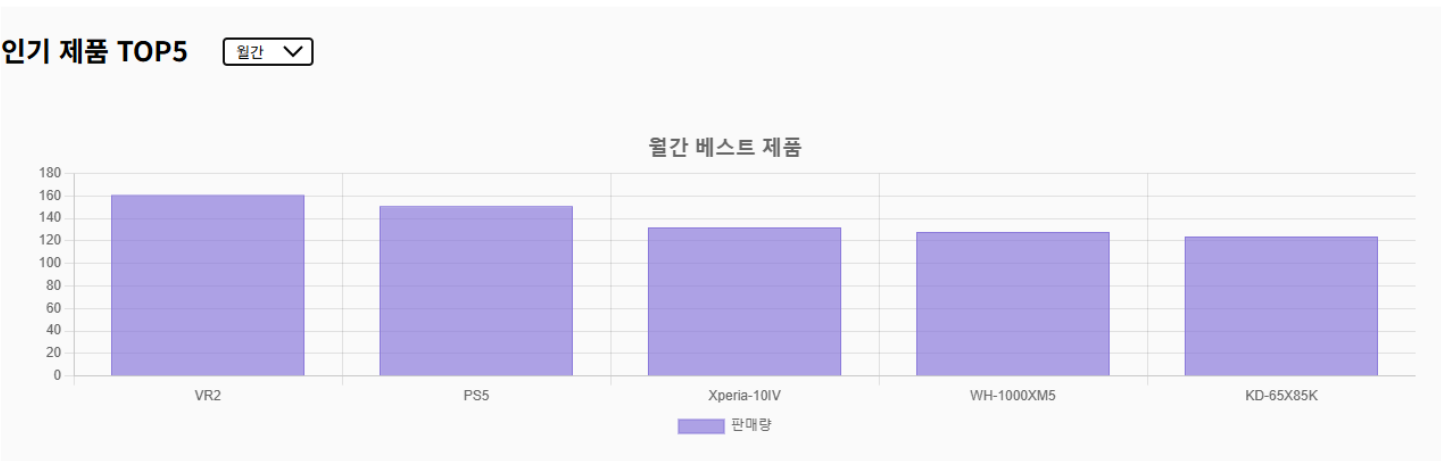
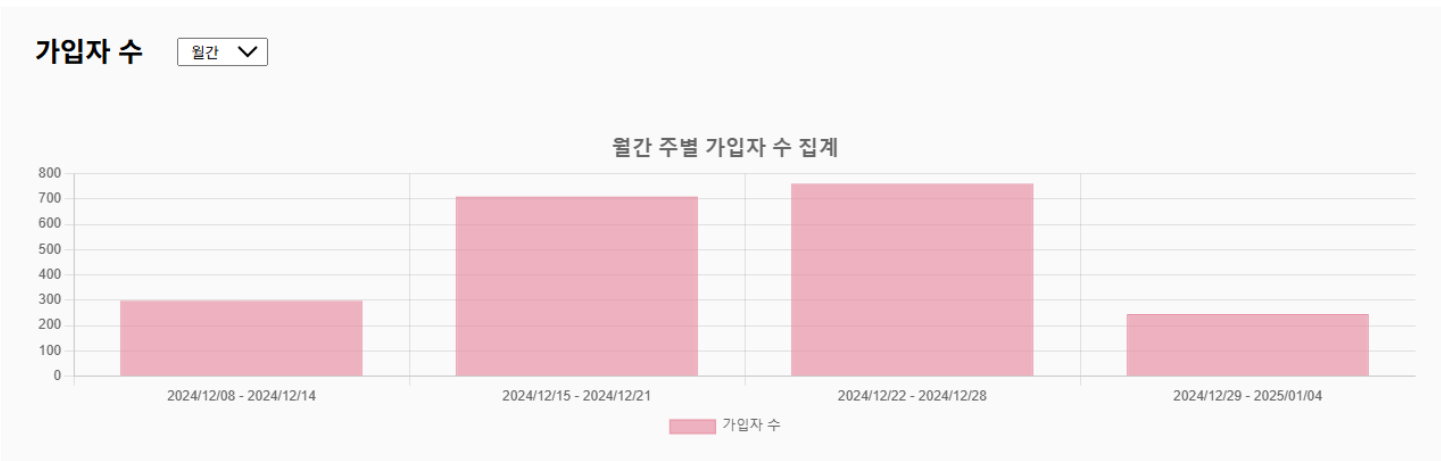
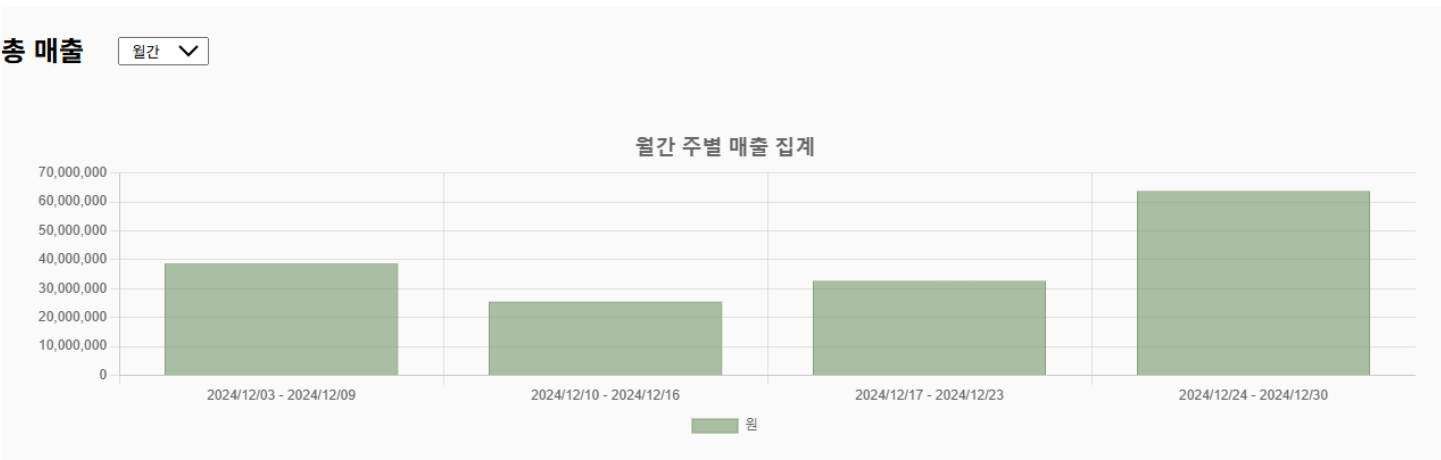
주간

DASHBOARD



월간

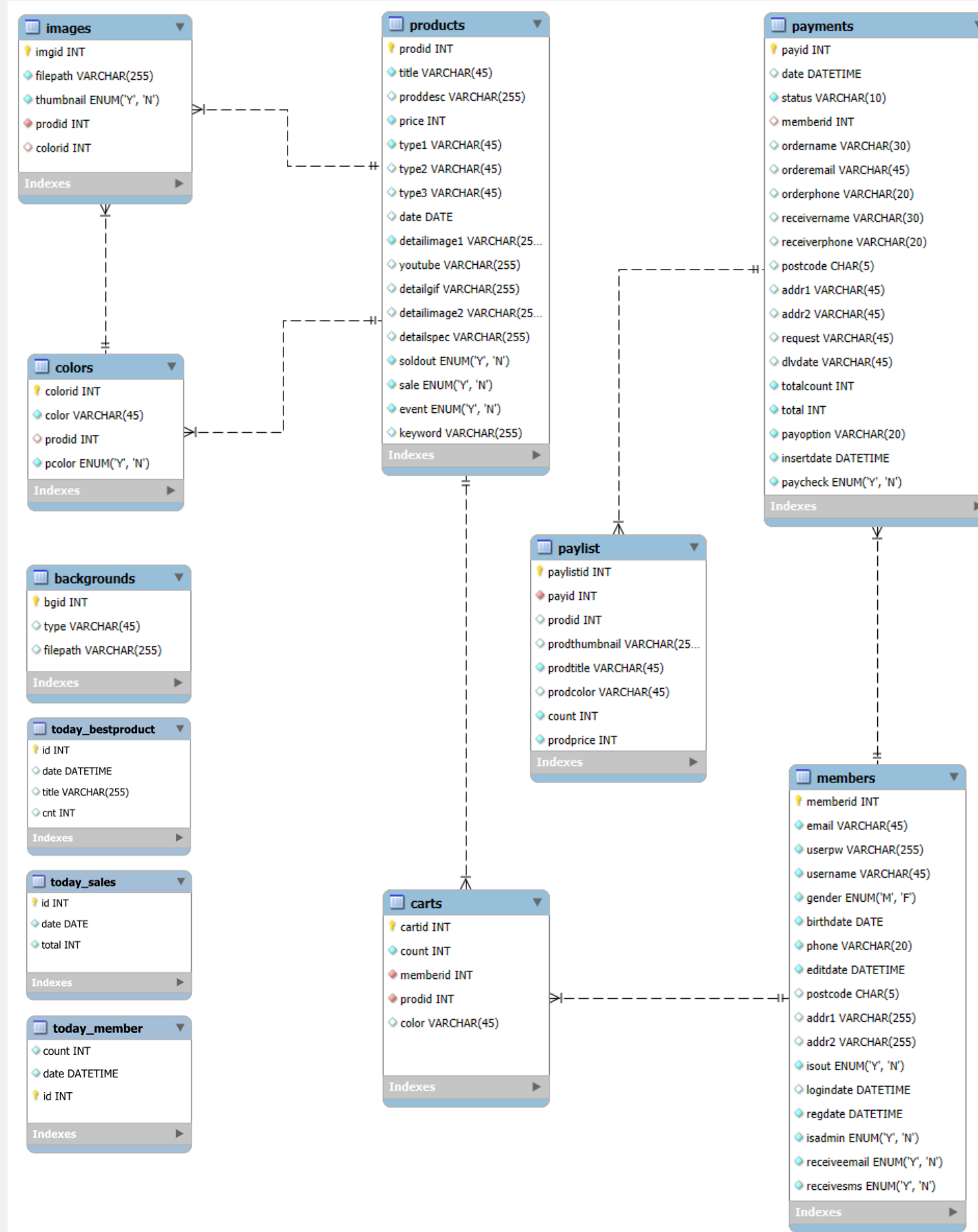
DASHBOARD



PART 2

# 데이터베이스

## 테이블 구성 (ERD)





# 테이블 명세서

| 제품

TableName		products				
Description		제품				
No	FieldName	DataType	Null	Key	Extra	Comment
1	prodid	int	NOT NULL	PRI	auto_increment	일련번호
2	title	varchar(45)	NOT NULL			제품 이름
3	proddesc	varchar(255)	NULL			제품 설명
4	price	int	NOT NULL			제품 가격
5	type1	varchar(45)	NOT NULL			제품 카테고리1
6	type2	varchar(45)	NULL			제품 카테고리2
7	type3	varchar(45)	NULL			제품 카테고리3
8	date	date	NULL			출시일
9	detailimage1	varchar(255)	NOT NULL			제품 상세의 이미지 경로1
10	youtube	varchar(255)	NULL			제품 상세의 유튜브 경로
11	detailgif	varchar(255)	NULL			제품 상세의 gif 경로
12	detailimage2	varchar(255)	NULL			제품 상세의 이미지 경로2
13	detailspec	varchar(255)	NULL			제품 상세의 스펙
14	soldout	enum('Y','N')	NOT NULL			제품의 품절여부 (Y/N)
15	sale	enum('Y','N')	NOT NULL			제품의 할인 유무 (Y/N)
16	event	enum('Y','N')	NOT NULL			제품의 기획전 유무 (Y/N)
17	keyword	varchar(255)	NULL			검색 결과 키워드

| 장바구니

TableName		carts				
Description		장바구니				
No	FieldName	DataType	Null	Key	Extra	Comment
1	cartid	int	NOT NULL	PRI	auto_increment	일련번호
2	count	int	NOT NULL			수량
3	memberid	int	NOT NULL	MUL		회원의 일련번호
4	prodid	int	NOT NULL	MUL		제품의 일련번호
5	color	varchar(45)	NULL			제품의 색상

| 배경 이미지

TableName		backgrounds				
Description		배경이미지 테이블				
No	FieldName	DataType	Null	Key	Extra	Comment
1	bgid	int	NOT NULL	PRI	auto_increment	
2	type	varchar(45)	NULL			제품의 타입
3	filepath	varchar(255)	NULL			파일 경로

| 색상

TableName		colors				
Description		색상				
No	FieldName	DataType	Null	Key	Extra	Comment
1	colorid	int	NOT NULL	PRI	auto_increment	일련번호
2	color	varchar(45)	NOT NULL			색상
3	prodid	int	NULL	MUL		상품의 일련번호
4	pcolor	enum('Y','N')	NOT NULL			기본색상

| 이미지

TableName		images				
Description		제품 이미지				
No	FieldName	DataType	Null	Key	Extra	Comment
1	imgid	int	NOT NULL	PRI	auto_increment	일련번호
2	filepath	varchar(255)	NOT NULL			이미지 파일 경로
3	thumbnail	enum('Y','N')	NOT NULL			제품 대표 이미지 (Y/N)
4	prodid	int	NOT NULL	MUL		제품의 일련번호
5	colorid	int	NULL	MUL		색상의 일련번호



# 테이블 명세서

## | 결제 목록

TableName	paylist					
Description	결제 제품 목록					
No	FieldName	DataType	Null	Key	Extra	Comment
1	paylistid	int	NOT NULL	PRI	auto_increment	일련번호
2	payid	int	NOT NULL	MUL		결제내역의 일련번호
3	prodid	int	NULL			제품의 일련번호
4	prodthumbnail	varchar(255)	NULL			제품의 대표 이미지 경로
5	prodtitle	varchar(45)	NOT NULL			제품의 이름
6	prodcolor	varchar(45)	NULL			구매한 제품의 색상
7	count	int	NOT NULL			제품의 수량
8	prodprice	int	NOT NULL			제품의 가격

## | 결제

TableName	payments					
Description	결제					
No	FieldName	DataType	Null	Key	Extra	Comment
1	payid	int	NOT NULL	PRI	auto_increment	일련번호
2	date	datetime	NULL			결제날짜
3	status	varchar(10)	NOT NULL			처리상태
4	memberid	int	NULL	MUL		회원의 일련번호
5	ordername	varchar(30)	NULL			주문자 이름
6	orderemail	varchar(45)	NULL			
7	orderphone	varchar(20)	NULL			주문자 휴대폰번호
8	receivername	varchar(30)	NULL			수령인 이름
9	receiverphone	varchar(20)	NULL			수령인 휴대폰번호
10	postcode	char(5)	NULL			우편번호
11	addr1	varchar(45)	NULL			검색된 주소
12	addr2	varchar(45)	NULL			상세 주소
13	request	varchar(45)	NULL			배송 요청 사항
14	dlvdate	varchar(45)	NULL			배송일
15	totalcount	int	NOT NULL			전체 수량
16	total	int	NOT NULL			총 결제금액
17	payoption	varchar(20)	NOT NULL			결제방법
18	insertdate	datetime	NOT NULL			구매 희망 일시
19	paycheck	enum('Y','N')	NOT NULL			결제유무

## | 회원 정보

TableName	members					
Description	회원정보					
No	FieldName	DataType	Null	Key	Extra	Comment
1	memberid	int	NOT NULL	PRI	auto_increment	일련번호
2	email	varchar(45)	NOT NULL			이메일
3	userpw	varchar(255)	NOT NULL			비밀번호
4	username	varchar(45)	NOT NULL			이름
5	gender	enum('M','F')	NOT NULL			성별 (M/F)
6	birthdate	date	NOT NULL			생년월일
7	phone	varchar(20)	NOT NULL			휴대폰번호
8	editdate	datetime	NOT NULL			변경일시
9	postcode	char(5)	NULL			우편번호
10	addr1	varchar(255)	NULL			검색된 주소
11	addr2	varchar(255)	NULL			상세 주소
12	isout	enum('Y','N')	NOT NULL			탈퇴 여부 (Y/N)
13	logindate	datetime	NULL			마지막 로그인 일시
14	regdate	datetime	NOT NULL			등록일시
15	isadmin	enum('Y','N')	NOT NULL			관리자 여부 (Y/ N)
16	receivemail	enum('Y','N')	NOT NULL			광고성 정보 이메일 수신 여부 (Y/N)
17	receivesms	enum('Y','N')	NOT NULL			광고성 정보 문자 수신 여부 (Y/N)

## | 인기제품순위

TableName	today_product					
Description	인기제품순위					
No	FieldName	DataType	Null	Key	Extra	Comment
1	id	int	NOT NULL	PRI	auto_increment	일련번호
2	date	datetime	NULL			구매날짜
3	title	varchar(255)	NULL			제품명
4	cnt	int	NULL			개수

## | 날짜별가입자수

TableName	today_member					
Description	날짜별 가입자 수					
No	FieldName	DataType	Null	Key	Extra	Comment
1	id	int	NOT NULL	PRI	auto_increment	집계 번호
2	date	datetime	NOT NULL			가입한 날짜
3	count	int	NOT NULL			가입자 수

## | 날짜별매출

TableName	today_sales					
Description	날짜별 매출					
No	FieldName	DataType	Null	Key	Extra	Comment
1	id	int	NOT NULL	PRI	auto_increment	일련번호
2	date	date	NOT NULL			날짜
3	count	int	NOT NULL			매출

PART 3

테스트

## 단위 테스트 (MyBatis)

```

@Test
@DisplayName("일별 매출 집계 조회 테스트")
void selectListTest() {
    // TodaySale input = new TodaySale();
    List<TodaySale> output = todaySaleMapper.selectList(-7);
    log.debug("output: " + output);
}

```

id	date	total
76	2024-12-20	9470000
75	2024-12-21	1490000
74	2024-12-22	12160000
66	2024-12-23	2690000
72	2024-12-24	0
109	2024-12-25	7590000
127	2024-12-26	28437000

```

@Test
@DisplayName("주간 매출 집계 테스트")
void selectWeekListTest() {
    List<TodayMember> output = todayMapper.selectWeekMemberCount();
    log.debug("output: " + output);
}

```

date	count
2024-12-20 00:00:00	110
2024-12-21 00:00:00	130
2024-12-22 00:00:00	150
2024-12-23 00:00:00	140
2024-12-24 00:00:00	105
2024-12-25 00:00:00	110
2024-12-26 00:00:00	120

```

@Test
@DisplayName("주간 베스트 상품 검색 테스트")
void selectWeeklyList() {
    List<Today_BestProduct> input = today_BestProductMapper.selectWeeklyList(input:null);
    for (Today_BestProduct item : input) {
        log.debug("item: " + item.toString());
    }
}

```

title	cnt
XR-55A80K	91
Xperia-10IV	59
ILCE-7M4	42
PS5	34
WH-1000XM5	5

## 단위 테스트 (Service)

```

@Test
@DisplayName("일별 매출 조회 테스트")
void getList() {
    List<TodaySale> output = null;
    int day = 7;
    try {
        output = todaySaleService.getList(day);
    } catch (Exception e) {
        log.error(msg:"서비스 에러", e);
    }

    log.debug("output: " + output);
}

```

id	date	total
76	2024-12-20	9470000
75	2024-12-21	1490000
74	2024-12-22	12160000
66	2024-12-23	2690000
72	2024-12-24	0
109	2024-12-25	7590000
127	2024-12-26	28437000

```

@Test
@DisplayName("주간 신규 회원 조회 테스트")
void getWeekList() {
    List<TodayMember> output = null;

    try {
        output = todayService.selectWeekMemberCount();
    } catch (Exception e) {
        log.error(msg:"서비스 에러", e);
    }

    log.debug("output: " + output);
}

```

date	count
2024-12-20 00:00:00	110
2024-12-21 00:00:00	130
2024-12-22 00:00:00	150
2024-12-23 00:00:00	140
2024-12-24 00:00:00	105
2024-12-25 00:00:00	110
2024-12-26 00:00:00	120

```

@Test
@DisplayName("주간 베스트 상품 조회 테스트")
void getWeeklyList() {
    List<Today_BestProduct> output = null;

    try {
        output = today_BestProductService.selectWeeklyList(input:null);
    } catch (Exception e) {
        log.error(msg:"서비스 에러", e);
    }

    log.debug("output: " + output);
}

```

title	cnt
XR-55A80K	91
Xperia-10IV	59
ILCE-7M4	42
PS5	34
WH-1000XM5	5

# 단위 테스트 (Restful 테스트 - Thunderclient)

총 매출

GET ☐ http://localhost:8080/api/today\_sales/day

Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

Status: 200 OK Size: 260 Bytes Time: 37 ms

Response Headers<sup>8</sup> Cookies Results Docs

```
1 {
2   "timestamp": "2024-12-30T11:39:29.260735",
3   "status": 200,
4   "message": "OK",
5   "item": [
6     {
7       "id": 66,
8       "date": "2024-12-23",
9       "total": 2690000
10    },
11    {
12      "id": 72,
13      "date": "2024-12-24",
14      "total": 0
15    },
16    {
17      "id": 109,
18      "date": "2024-12-25",
19      "total": 7590000
20    },
21    {
22      "id": 127,
23      "date": "2024-12-26"
```

신규 가입자 수

GET ☐ http://localhost:8080/api/week\_member

Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

Status: 200 OK Size: 282 Bytes Time: 454 ms

Response Headers<sup>8</sup> Cookies Results Docs

```
1 {
2   "timestamp": "2024-12-30T15:02:59.646855800",
3   "status": 200,
4   "message": "OK",
5   "item": [
6     {
7       "id": 0,
8       "date": "2024-12-23 00:00:00",
9       "count": 140
10    },
11    {
12      "id": 0,
13      "date": "2024-12-24 00:00:00",
14      "count": 105
15    },
16    {
17      "id": 0,
18      "date": "2024-12-25 00:00:00",
19      "count": 110
20    },
21    {
22      "id": 0,
23      "date": "2024-12-26 00:00:00",
```

인기 상품 순위

GET ☐ http://localhost:8080/api/today\_best\_products


Query Headers<sup>2</sup> Auth Body Tests Pre Run

Status: 200 OK Size: 587 Bytes Time: 348 ms

Response Headers<sup>8</sup> Cookies Results Docs

```
1 {
2   "timestamp": "2024-12-31T17:45:23.309031700",
3   "status": 200,
4   "message": "OK",
5   "weekly": [
6     {
7       "id": 0,
8       "title": "XR-55A80K",
9       "date": null,
10      "cnt": 91
11    },
12    {
13      "id": 0,
14      "title": "Xperia-10IV",
15      "date": null,
16      "cnt": 59
17    },
18    {
19      "id": 0,
20      "title": "ILCE-7M4",
21      "date": null,
```

# 단위 테스트 (Restful 테스트 - Swagger)

 **Swagger**  
Supported by SMARTBEAR

/v3/api-docs

Explore

## SonyStore Swagger

1.0.0 OAS 3.0

/v3/api-docs

SonyStore REST API

Servers

http://localhost:8080 - Generated server url

### New Member API

주간, 월간 신규 가입자 수 API

GET

/api/week\_member

주간 일별 신규 회원 조회

GET

/api/month\_member

월간 주별 신규 회원 조회

### Sale API

주간, 월간 총 매출 API

GET

/api/today\_sales/day

총 매출 조회

### Best Product API

주간, 월간 인기 상품 순위 API

GET

/api/today\_best\_products

인기 상품 순위 조회

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/today_sales/day?day=28' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/api/today_sales/day?day=28
```

Server response

Code	Details
200	<div><div>Response body</div><pre>{   "timestamp": "2024-12-30T10:35:47.918681600",   "status": 200,   "message": "OK",   "item": [     {       "id": 99,       "date": "2024-12-02",       "total": 12160000     },     {       "id": 100,       "date": "2024-12-03",       "total": 9470000     },     {       "id": 101,       "date": "2024-12-04",       "total": 0     },     {       "id": 102,       "date": "2024-12-05",       "total": 3409000     },     {       "id": 103,       "date": "2024-12-06",       "total": 12160000     }   ] }</pre></div>

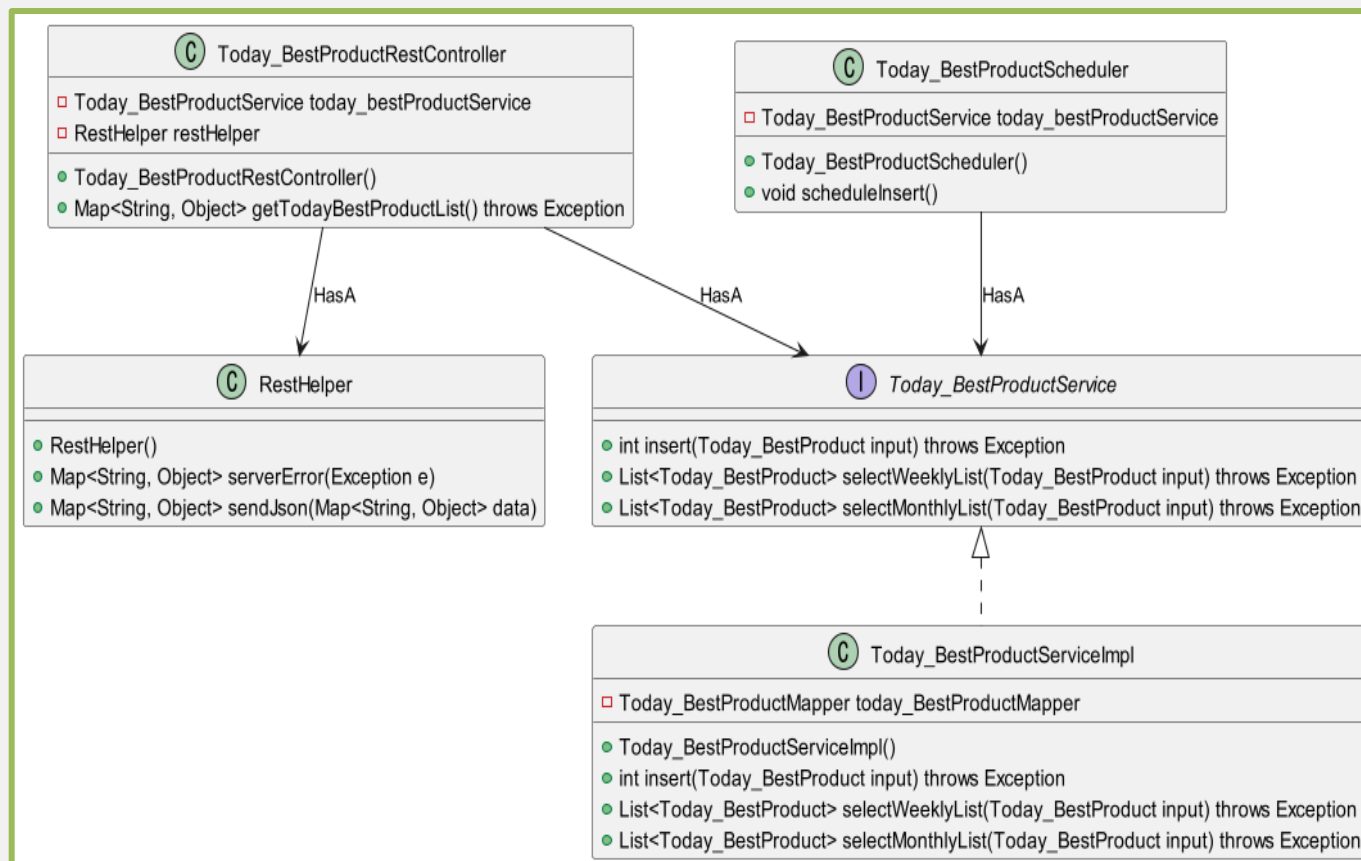
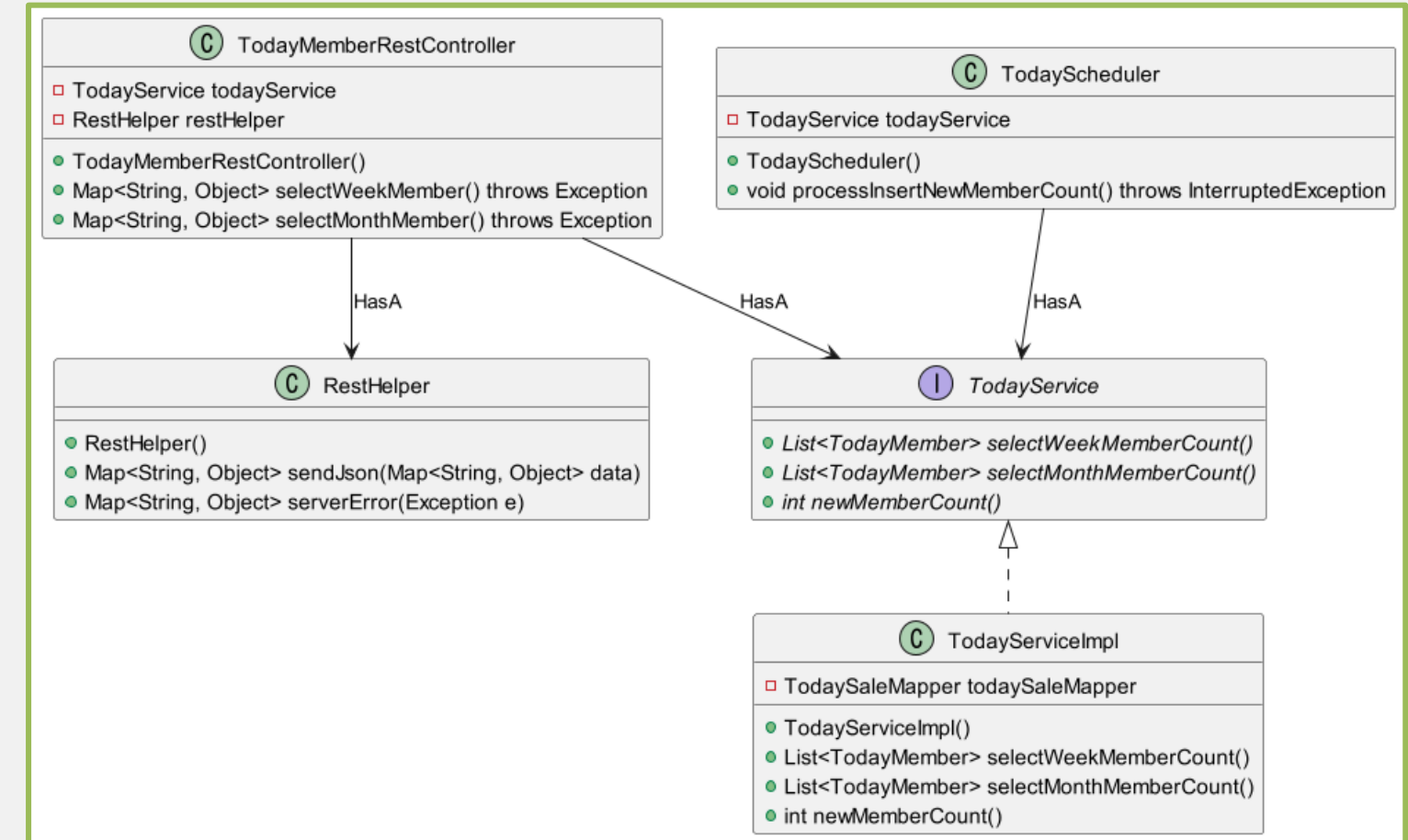
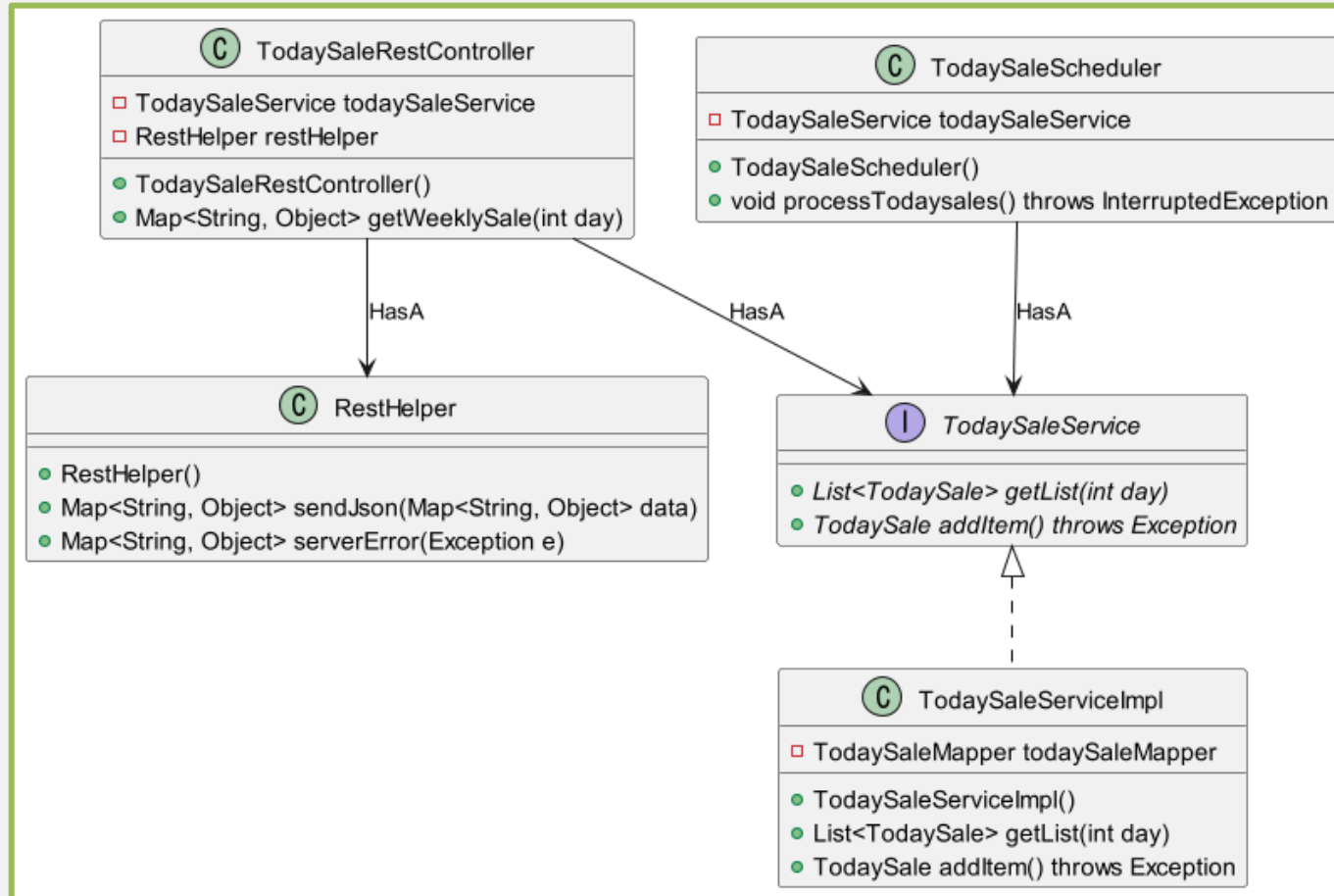
Download

PART 4

# 구현 기능



# 클래스 다이어그램



TodaySaleRestController : 총 매출을 조회하는 RESTful API 제공

TodaySaleScheduler : 매일 새벽 1시에 전날 매출 데이터를 집계하는 스케줄러

TodaySaleService : 전날 매출을 추가하고, 지정 기간 동안의 매출 데이터 조회

TodayMemberRestController : 주간 및 월간 신규 회원을 조회하는 RESTful API 제공

TodayScheduler : 매일 새벽 1시에 전날 신규 회원 데이터를 집계하는 스케줄러

TodayService : 신규 회원 데이터 추가 및 조회

TodayBestProductRestController : 주간 및 월간 상품 판매량을 조회하는 RESTful API 제공

TodayBestProductScheduler : 매일 새벽 1시에 전날 판매량을 집계하는 스케줄러

TodayBestProductService : 전날 판매량을 추가하고, 주별, 월별 판매량 데이터 조회

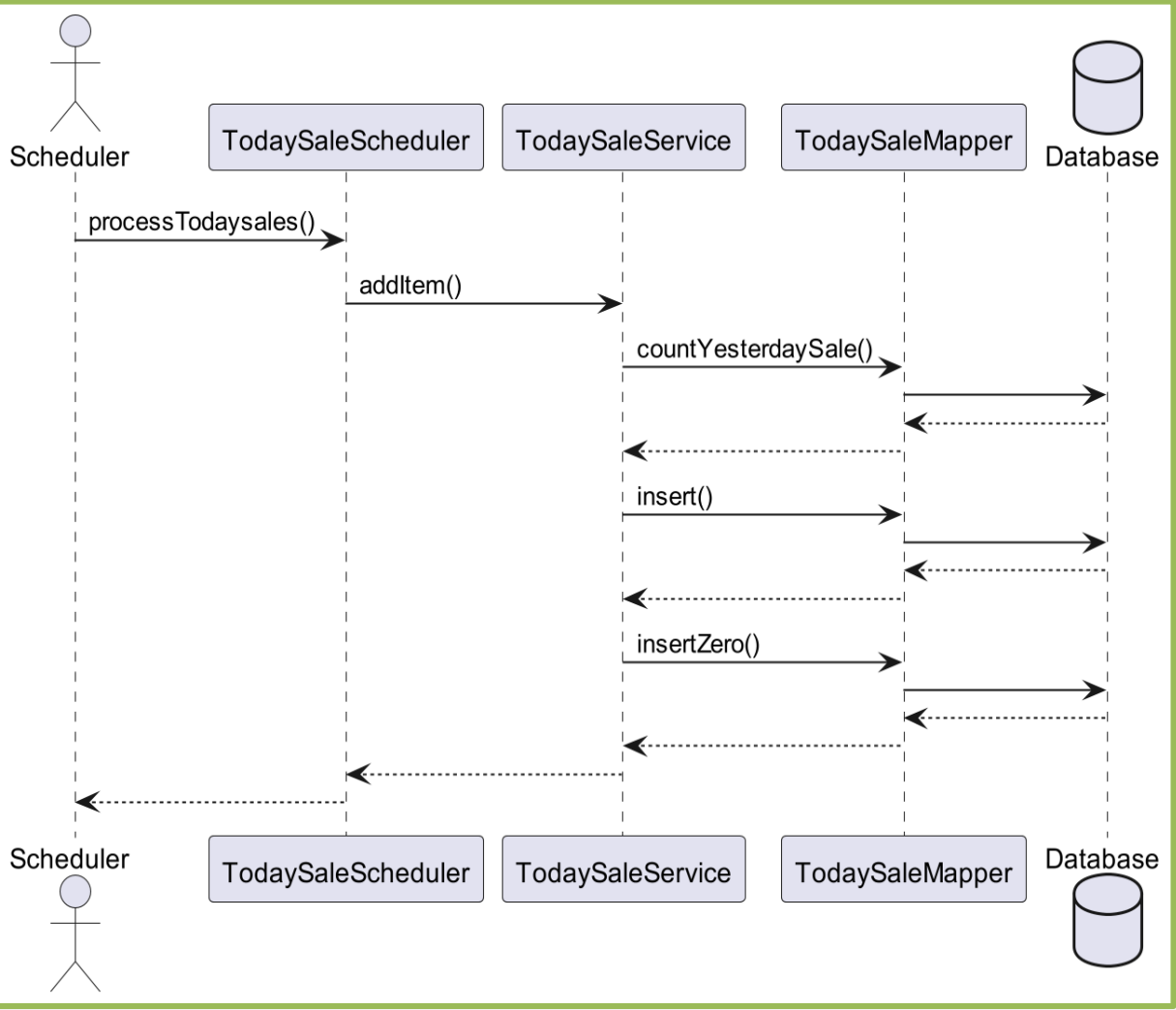
RestHelper : HTTP 응답을 설정하고 JSON 데이터를 반환하는 기능 제공

4-2

# 구현 기능 - 스케줄러

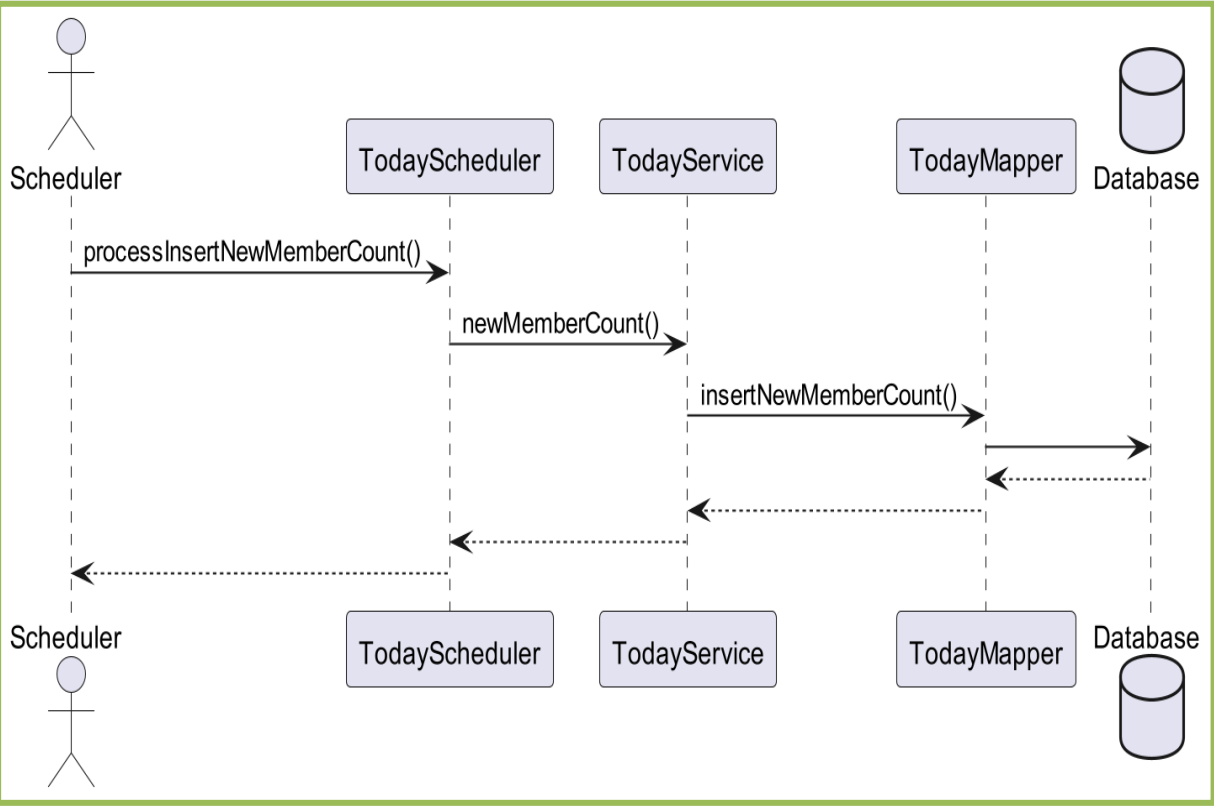
## 총 매출

매일 오전 1시에 전날 매출을 집계하여  
today\_sales 테이블에 저장



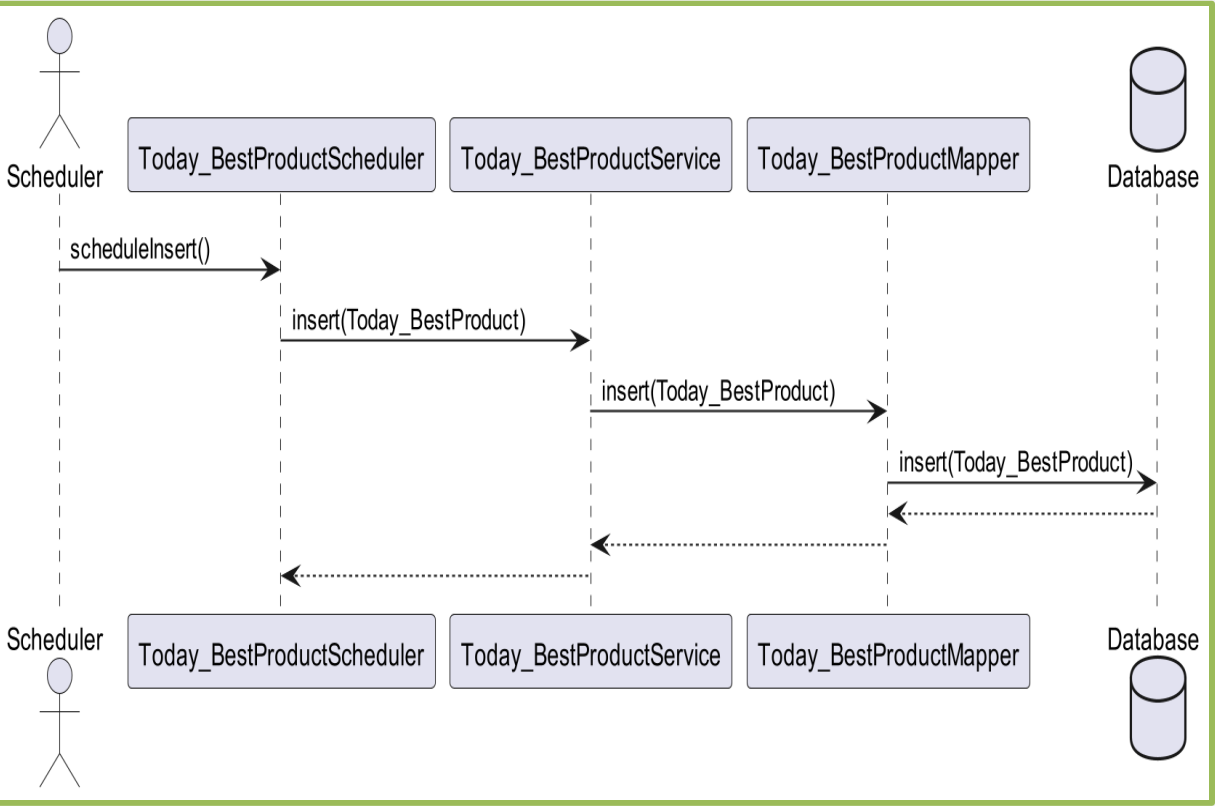
## 신규 가입자 수

매일 오전 1시에 등록 일자가 하루 전인 회원의 수와 날짜를  
today\_member 테이블에 저장

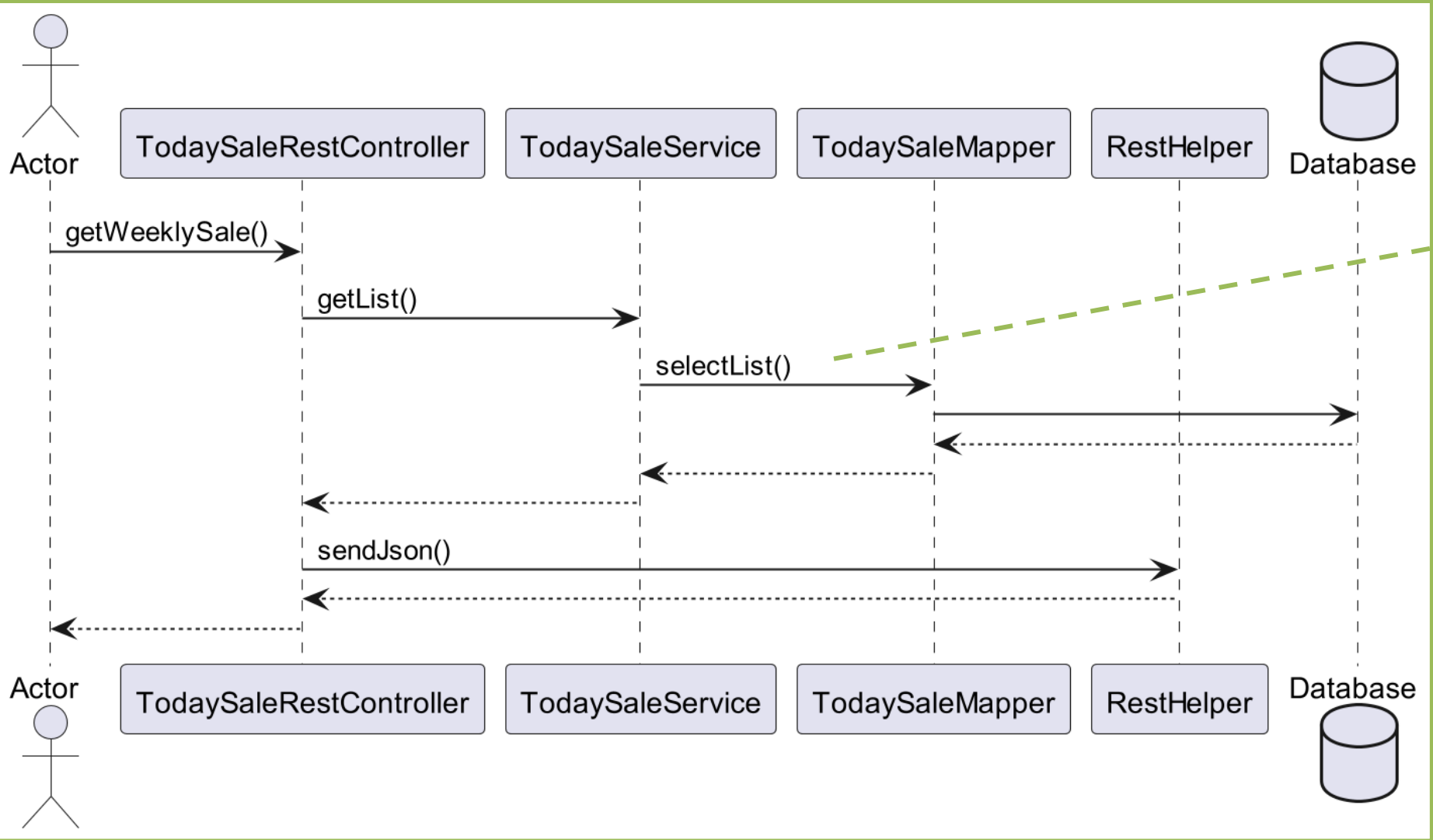


## 판매량

매일 오전 1시에 전날 판매량을 집계하여  
today\_bestproduct 테이블에 저장



# 4-3 구현 기능 - 총 매출 집계 (백엔드)



Mapper

```
/**
 * 일별 매출 집계 조회
 * @return 조회된 데이터 리스트
 */
@Select(
    "SELECT id, date, total FROM today_sales \n" +
    "WHERE date BETWEEN DATE(DATE_ADD(NOW(), INTERVAL #{day} DAY)) \n" +
    "AND DATE(DATE_ADD(NOW(), INTERVAL -1 DAY)) \n" +
    "ORDER BY date"
)
public List<TodaySale> selectList(int day);
```

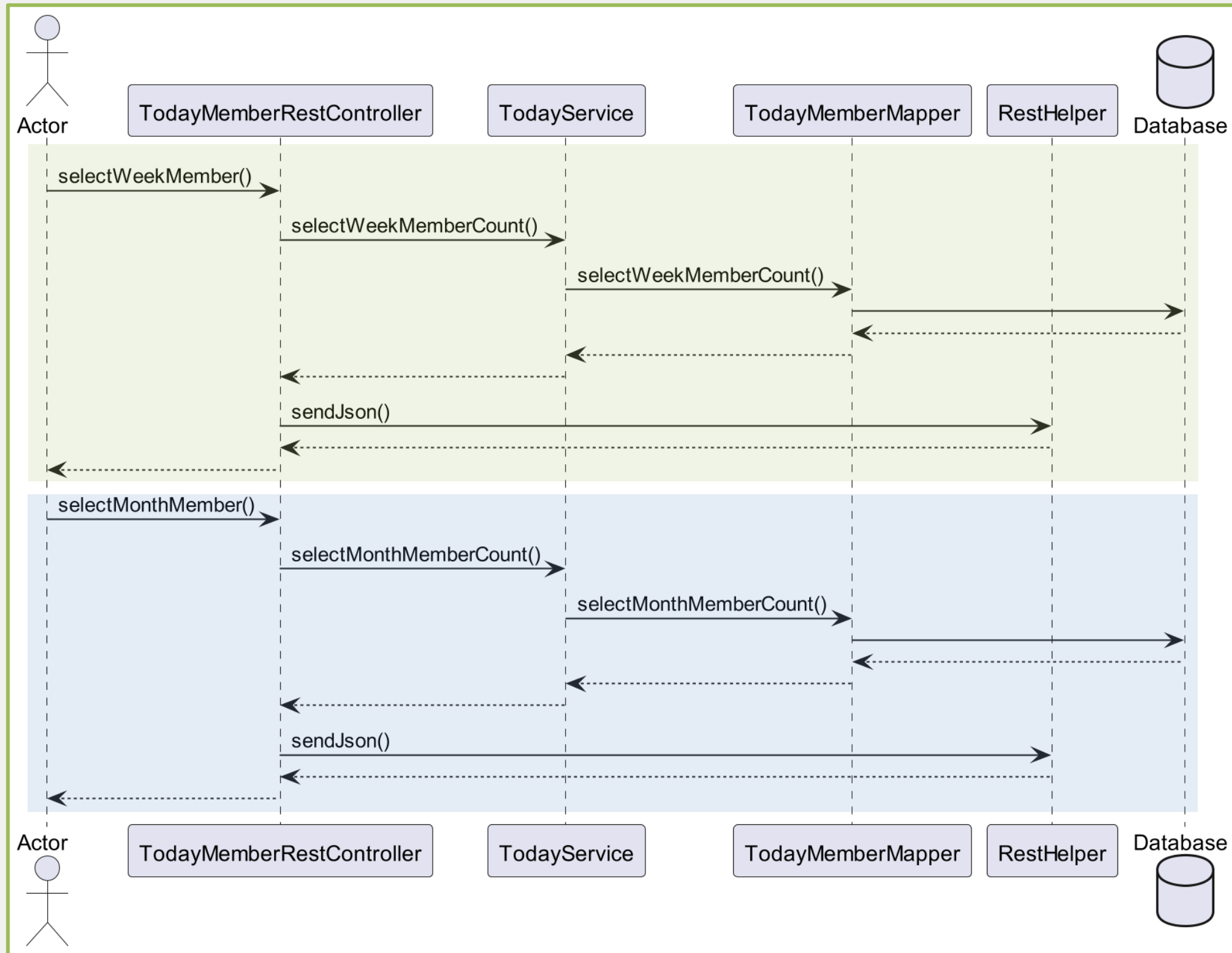
RestController

```
@Parameters({
    @Parameter(name="day", description = "조회할 일자", required = false)
})
public Map<String, Object> getWeeklySale(
    @RequestParam(value = "day", defaultValue = "7", required = false) int day
) {
```

요청받은 쿼리 파라미터에 따라 주간 또는 월간 데이터 조회

## 4-3

## 구현 기능 - 신규 회원 집계 (백엔드)



// 가입인원수 집계 후 테이블에 넣기

```
@Insert("insert into today_member (date, count) " +
        "select DATE(regdate) date, COUNT(*) count " +
        "from members " +
        "where DATE(regdate) = DATE(DATE_ADD(NOW(), INTERVAL -1 DAY)) " +
        "group by date")
```

```
public int insertNewMemberCount();
```

// 7간 일간 통계

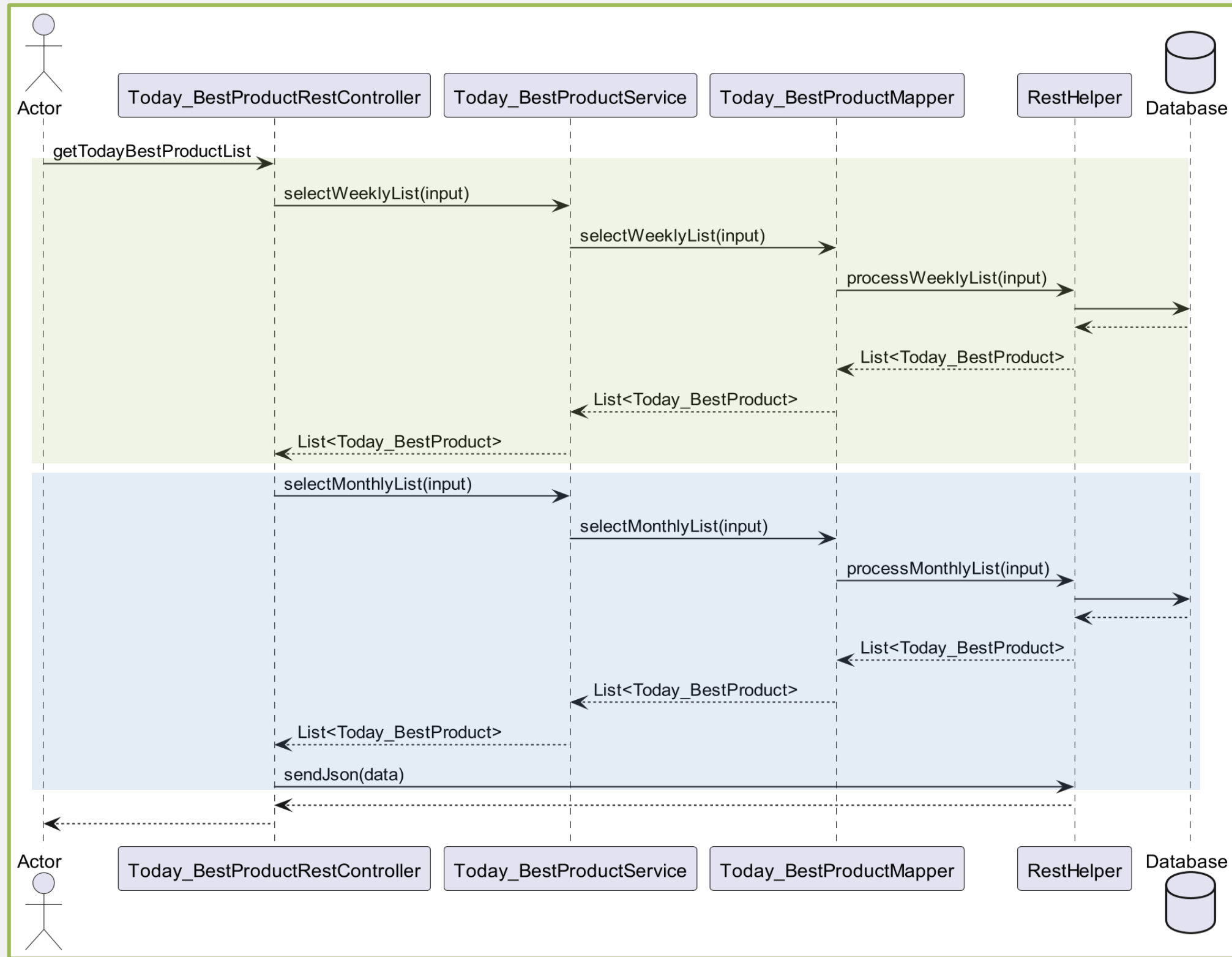
```
@Select("select date, count " +
        "from today_member " +
        "where date >= DATE(DATE_ADD(NOW(), INTERVAL -7 DAY))")
public List<TodayMember> selectWeekMemberCount();
```

// 한달간의 주간 통계

```
@Select("select date, SUM(count) as count " +
        "from (select concat( " +
        "DATE_FORMAT(" +
        "    DATE_SUB(`date`, INTERVAL (DAYOFWEEK(`date`)-1) DAY), " +
        "    '%Y/%m/%d'), " +
        "' - ', " +
        "DATE_FORMAT(" +
        "    DATE_SUB(`date`, INTERVAL (DAYOFWEEK(`date`)-7) DAY), " +
        "    '%Y/%m/%d') " +
        ") as date, count from today_member group by date, count " +
        "order by date desc limit 21) as today_member " +
        "group by date " +
        "order by date asc")
public List<TodayMember> selectMonthMemberCount();
```



## 4-3 구현 기능 - 상품별 판매량 집계 (백엔드)



```

// 집계 결과를 테이블에 바로 넣기
@Insert("INSERT INTO today_bestproduct (title, date, cnt) " +
        "SELECT pl.prodtitle, DATE(pm.date) as dt, COUNT(*) AS cnt " +
        "FROM playlist pl " +
        "INNER JOIN payments pm ON pl.payid = pm.payid " +
        "WHERE DATE(pm.date) = DATE(DATE_ADD(NOW(), INTERVAL 0 DAY)) " +
        "GROUP BY DATE(pm.date), pl.prodtitle " +
        "ORDER BY cnt DESC " +
        "LIMIT 10")
@Options(useGeneratedKeys = true, keyProperty = "id", keyColumn = "id")
public int insert(Today_BestProduct input);

// 주별 집계 결과 검색
@Select("SELECT title, SUM(cnt) as cnt " +
        "FROM today_bestproduct " +
        "WHERE date BETWEEN DATE_SUB(CURDATE(), INTERVAL 7 DAY) AND CURDATE() " +
        "GROUP BY title " +
        "ORDER BY cnt DESC " +
        "LIMIT 5")
@Results(id = "today_bestproductMap", value = {
    @Result(property = "title", column = "title"),
    @Result(property = "cnt", column = "cnt")
})
public List<Today_BestProduct> selectWeeklyList(Today_BestProduct input);

// 월별 집계 결과 검색
@Select("SELECT title, SUM(cnt) as cnt " +
        "FROM today_bestproduct " +
        "WHERE date BETWEEN DATE_SUB(CURDATE(), INTERVAL 30 DAY) AND CURDATE() " +
        "GROUP BY title " +
        "ORDER BY cnt DESC " +
        "LIMIT 5")
@Results(id = "weekly_bestproductMap", value = {
    @Result(property = "title", column = "title"),
    @Result(property = "cnt", column = "cnt")
})
public List<Today_BestProduct> selectMonthlyList(Today_BestProduct input);
  
```

## Redux Slice

```
export const getList = createAsyncThunk('SaleSlice/getList', async ( {url}, {rejectWithValue}) =>{
  let result = null;
  let args = { _sort: 'id', _order: 'desc' };

  try {
    result = await axiosHelper.get(url, args);
  } catch(err) {
    result = rejectWithValue(err);
  }

  return result;
});

const SaleSlice = reduxHelper.getDefaultSlice('SaleSlice', [getList]);

export default SaleSlice.reducer;
```

Axios를 통해 백엔드로부터 데이터를 수신

## React Component

```
useEffect(() => {
  const url = period === 'weekly' ? '/api/today_sales/day' : '/api/today_sales/day?day=28';
  dispatch(getList({ url }));
}, [dispatch, period]);
```

혹을 사용하여 드롭다운의 값이 변경될 때마다  
“주간” 또는 “월간” 데이터 가져오기

Request URL	
http://localhost:8080/api/today_sales/day?day=7	
Server response	
Code	Details
200	<div>Response body</div> <pre>{   "timestamp": "2024-12-31T11:40:48.180578300",   "status": 200,   "message": "OK",   "item": [     {       "id": 72,       "date": "2024/12/24",       "total": 0     },     {       "id": 109,       "date": "2024/12/25",       "total": 7590000     },     {       "id": 127,       "date": "2024/12/26",       "total": 38137000     }   ] }</pre>

## RESTful API

백엔드로부터 받아온 집계 데이터를  
React 프론트엔드로 전달

Request URL	
http://localhost:8080/api/today_sales/day?day=28	
Server response	
Code	Details
200	<div>Response body</div> <pre>{   "timestamp": "2024-12-31T11:47:07.892213",   "status": 200,   "message": "OK",   "item": [     {       "id": 100,       "date": "2024/12/03",       "total": 9470000     },     {       "id": 101,       "date": "2024/12/04",       "total": 0     },     {       "id": 102,       "date": "2024/12/05",       "total": 3409000     }   ] }</pre>

## 4-3

## 구현 기능 - 대시보드 (프론트엔드)

예시 : 총 매출

React Component

```
const dispatch = useDispatch();
const {item} = useSelector( state => state.SaleSlice );
const [period, setPeriod] = useState('weekly');
const [data, setData] = useState({ keys: [], values: [] });
```

혹을 사용하여 Redux 상태를 관리

```
useEffect( () => {

  if (!item) return;

  let keys = [];
  let values = [];

  if (period === 'weekly') {
    keys = item.map(v => v.date);
    values = item.map(v => v.total);
  } else {
    const weeks = [];
    for (let i=0; i<item.length; i+=7) {
      weeks.push( item.slice( i, i+7 ) );
    }

    const weeklyData = weeks.map( week => week.reduce( (acc, cur) => acc + cur.total, 0 ) );

    keys = weeks.map( (v,i) => `${v[0].date} - ${v[v.length-1].date}` );
    values = weeklyData;
  }

  setData({ keys, values });

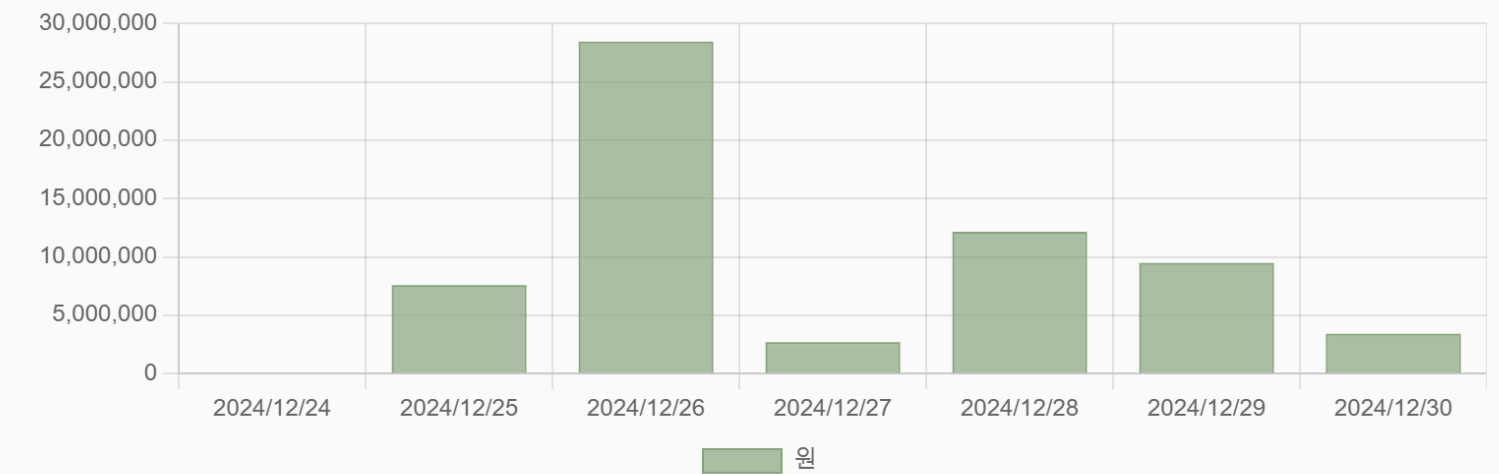
}, [item, period]);
```

주간, 월간 에 따라 집계 데이터를 반복문 처리하여 key와 value에 해당하는 값을 배열에 저장

총 매출

주간 ▼

주간 일별 매출 집계



총 매출

월간 ▼

월간 주별 매출 집계

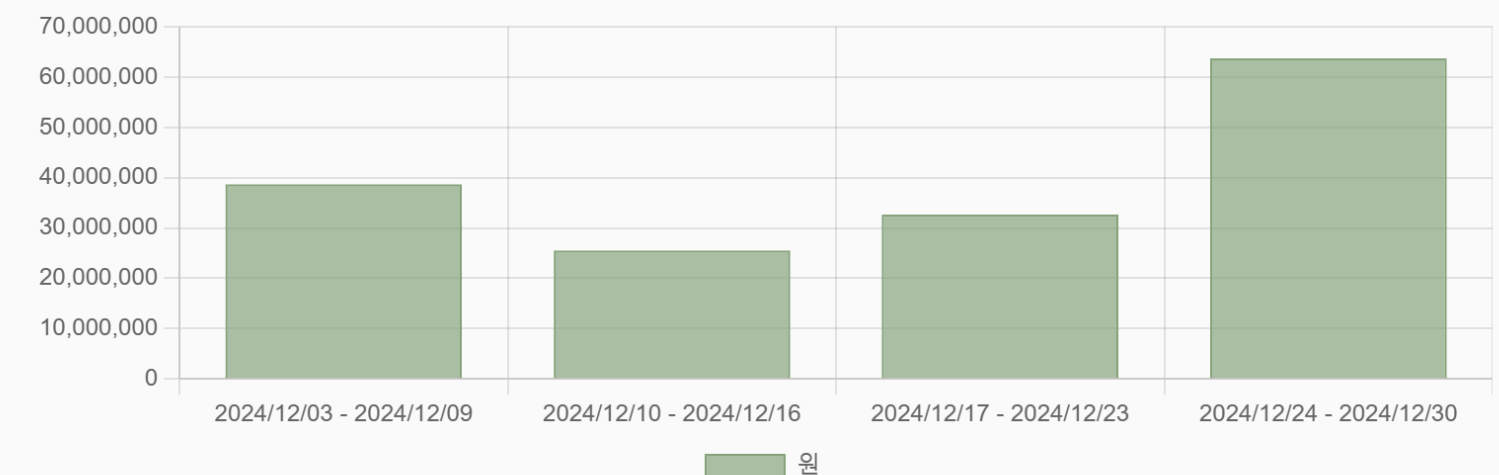


Chart.js 라이브러리를 이용한 그래프 구현



감사합니다