

한의원 관리자

제작기간: 2023-01-26 ~ 2023.02.06
버전:v0.0.1

강유준

목차

a table of contents

- 1 기획 의도
- 2 소프트웨어 구성
- 3 목업 디자인
- 4 Back-End
- 5 Front-End





Part 1

기획 의도

한의원 관리자 기획 의도

병원에서 사용할 고객 관리 API를 구상하던 중 한의학에 관심이 생겨 한의원 고객 관리 API를 구상하게 되었습니다.

한의원을 찾아온 고객 정보 리스트와 진료 기록을 쉽게 열람할 수 있게 만들고 진료 기록은 날짜와 환자 별로 검색할 수 있게 기획했습니다.

병원이라는 특수성에 급여 진료를 따로 계산하는 것도 기획에 넣었습니다.

Part 2

소프트웨어 구성



Back-End



- Java
- Spring Boot
- JPA
- Postgres

Front-End



- Javascript
- vue.js
- Nuxt



Part 3

목업 디자인

메인 페이지



신규 고객 등록 페이지

이름 *홍길동

주민등록번호 *970000-0000000

휴대폰 번호 *010-0000-0000

주소

비고

내용을 입력해주세요

등록

고객 관리 페이지

고객 관리

신규 등록 뒤로 가기

ID	이름	주민등록번호	휴대폰 번호	주소	첫 방문일	비고
1	홍길동	970000-1000000	010-0000-0000	경기도 안산시..	2023-01-25	허리 디스크
2	홍동길	710000-1000000	010-0000-0000	경기도 안산시..	2023-01-25	
3	홍영길	850000-1000000	010-0000-0000	경기도 안산시..	2023-01-25	
4	강영길	780000-1000000	010-0000-0000	경기도 안산시..	2023-01-25	목 디스크
5	강길동	890000-1000000	010-0000-0000	경기도 안산시..	2023-01-25	
6	강동길	010000-3000000	010-0000-0000	경기도 안산시..	2023-01-25	
7	김영서	980000-2000000	010-0000-0000	경기도 안산시..	2023-01-25	
8	김인수	860000-2000000	010-0000-0000	경기도 안산시..	2023-01-25	
9	고영희	720000-2000000	010-0000-0000	경기도 안산시..	2023-01-25	
10	고영려	990000-2000000	010-0000-0000	경기도 안산시..	2023-01-25	

고객 등록 완료

***님의 정보가 등록되었습니다.

돌아가기 추가 등록

고객 등록 완료 페이지

목업 디자인

진료 기록 관리 페이지

날짜별 환자별 진료비 기록

ID	고객ID	진료 내용	급여 여부	진료비	진료일	진료시간	정산 여부
1	1	주나 치료	비급여	50000	2023-01-30	10:00:08	완료
2	1	약침 치료	급여	3000	2023-01-30	10:00:08	완료
3	1	물리 치료	급여	7000	2023-01-30	10:00:08	완료
4	3						
5	5						
6	6						
7	9						
8	2						
9	4						
10	5						

진료 기록 관리 날짜 별 페이지

< prev

ID	이름	유대문 번호	주민등록번호	급여 여부	진료 내용	진료비	진료일	진료시간	정산 여부
1	홍길동	010-0000-0000	970000-1000000	비급여	주나 치료	50000	2023-01-30	10:00:08	완료
2	홍길동	010-0000-0000	970000-1000000	급여	약침 치료	3000	2023-01-30	10:00:08	완료
3	홍길동	010-0000-0000	970000-1000000	급여	물리 치료	7000	2023-01-30	10:00:08	완료
4									
5									
6									
7									
8									
9									
10									

< prev

고객ID	이름	주민등록번호	휴대폰 번호	주소	첫 방문일	비고	진료ID	진료내용	급여 여부	진료비	진료일	진료시간	정산 여부
1	홍길동	970000-1000000	010-0000-0000	경기도 안산 시...	2023-01-25	허리 디스크	1	주나 치료	비급여	50000	2023-01-30	10:00:08	완료
1	홍길동	970000-1000000	010-0000-0000	경기도 안산 시...	2023-01-25	허리 디스크	2	약침 치료	급여	3000	2023-01-30	10:00:08	완료
1	홍길동	970000-1000000	010-0000-0000	경기도 안산 시...	2023-01-25	허리 디스크	3	물리 치료	급여	7000	2023-01-30	10:00:08	완료
3													
5													
6													
9													
2													
4													
5													

< prev

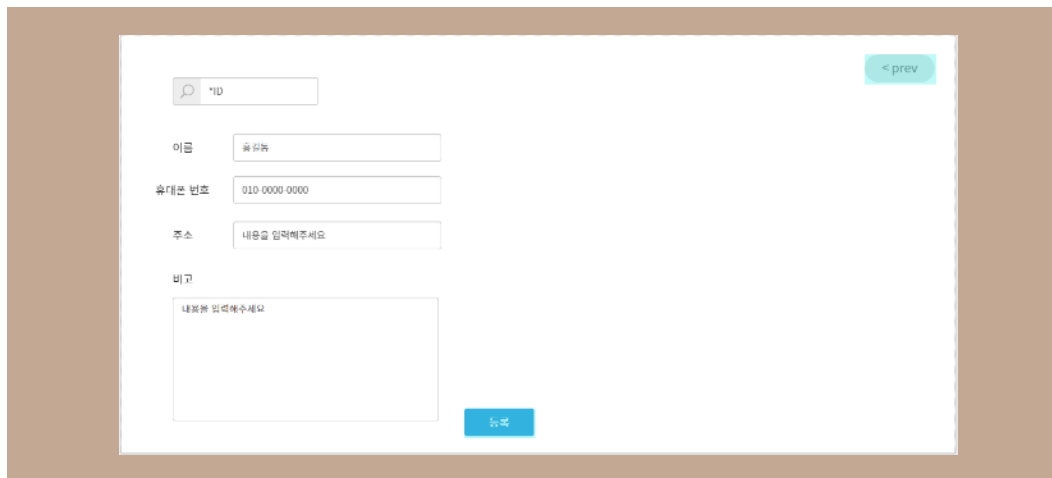
이름	주민등록번호	휴대폰 번호	진료내용	진료비	비급여시 비용	공단 청구비	진료일
홍길동	970000-1000000	010-0000-0000	약침 치료	3000	15000	12000	2023-01-30
홍길동	970000-1000000	010-0000-0000	물리 치료	7000	20000	13000	2023-01-30

진료 기록 관리 환자 별 페이지

진료비 기록 페이지

목업 디자인

고객 정보 수정 페이지



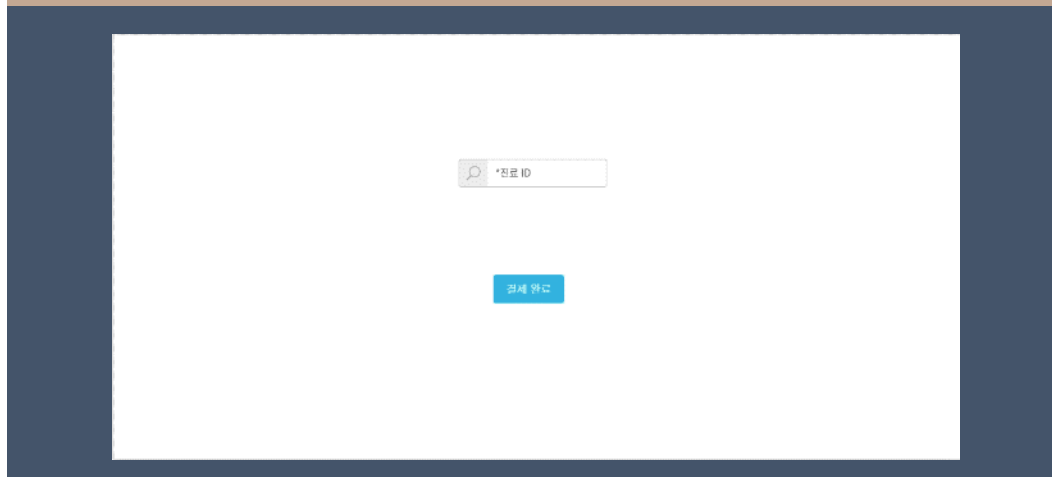
This mockup shows a form for modifying customer information. It includes a search bar with a magnifying glass icon and the text '*ID'. Below the search bar are four input fields: '이름' (Name) with the placeholder '홍길동', '휴대폰 번호' (Mobile Number) with the placeholder '010-0000-0000', '주소' (Address) with the placeholder '내용을 입력해주세요', and '비고' (Remarks) with the placeholder '내용을 입력해주세요'. A '< prev' button is located in the top right corner, and a '등록' (Register) button is at the bottom right.

고객 정보 수정 완료 페이지



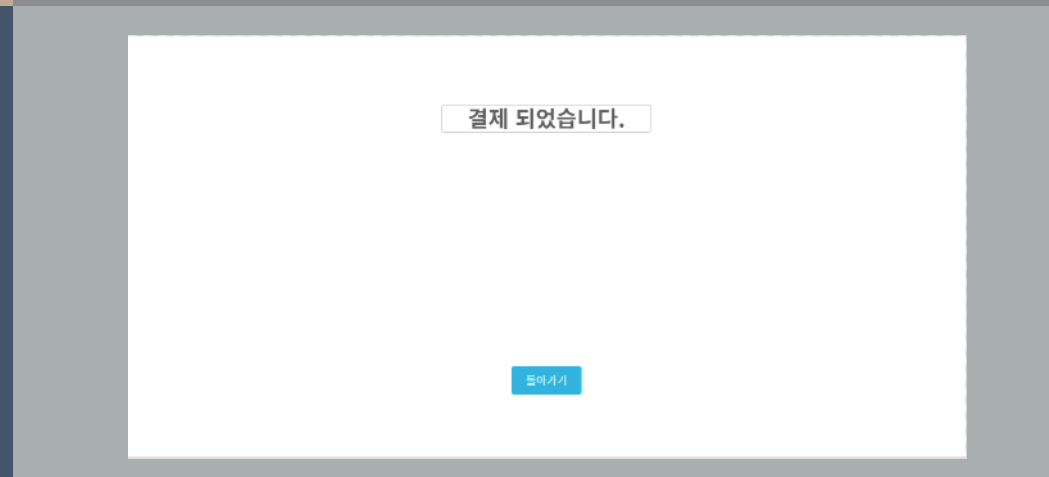
This mockup shows the completion screen for modifying customer information. It features the title '고객 정보 수정 완료' (Customer Information Modification Complete) and a message: '***님의 정보가 수정되었습니다.' (The information of *** has been modified). At the bottom, there are two buttons: '돌아가기' (Go Back) and '추가 수정' (Add Modification).

결제 페이지



This mockup shows a payment page with a search bar containing a magnifying glass icon and the text '*전로 ID'. A '결제 완료' (Payment Complete) button is centered at the bottom.

결제 완료 페이지



This mockup shows the completion screen for payment. It displays the message '결제 되었습니다.' (Payment has been completed) in a box. A '돌아가기' (Go Back) button is located at the bottom center.

목업 디자인

날짜 검색 페이지

This mockup shows a date search interface. It features a light gray search area with a white border, set against a brown background. Inside the search area, there are two input fields: the first is labeled '날짜' (Date) and the second is preceded by a magnifying glass icon. Below these fields is a blue button labeled '검색' (Search). The entire search area is enclosed in a brown border. At the bottom of the page, there is a dark blue footer bar.

환자 검색 페이지

This mockup shows a patient search interface. It features a light gray search area with a white border, set against a gray background. Inside the search area, there are two input fields: the first is labeled '이름' (Name) and the second is preceded by a magnifying glass icon. Below these fields is a blue button labeled '검색' (Search). The entire search area is enclosed in a gray border. The bottom of the page features a gray footer bar.

진료비 기록 검색 페이지

Part 4

Back-End



Back-End (Entity)

```

@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class HospitalPatient {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false, length = 20)
    private String patientName;
    @Column(nullable = false, length = 13)
    private String patientPhone;
    @Column(nullable = false, length = 14)
    private String registrationNumber;
    @Column(nullable = false, length = 100)
    private String address;
    @Column(nullable = false, columnDefinition = "TEXT")
    private String memo;
    @Column(nullable = false)
    private LocalDate visitFirst;

    public void putInfoUpdate(PatientInfoUpdateRequest request) {
        this.patientName = request.getPatientName();
        this.patientPhone = request.getPatientPhone();
        this.address = request.getAddress();
        this.memo = request.getMemo();
    }

    private HospitalPatient(HospitalPatientBuilder builder) {
        this.patientName = builder.patientName;
        this.patientPhone = builder.patientPhone;
        this.registrationNumber = builder.registrationNumber;
        this.address = builder.address;
        this.memo = builder.memo;
        this.visitFirst = builder.visitFirst;
    }

    public static class HospitalPatientBuilder implements
CommonModelBuilder<HospitalPatient> {
        private final String patientName;
        private final String patientPhone;
        private final String registrationNumber;
        private final String address;
        private final String memo;
        private final LocalDate visitFirst;

        public HospitalPatientBuilder(PatientRequest request) {
            this.patientName = request.getPatientName();
            this.patientPhone = request.getPatientPhone();
            this.registrationNumber = request.getRegistrationNumber();
            this.address = request.getAddress();
            this.memo = request.getMemo();
            this.visitFirst = LocalDate.now();
        }

        @Override
        public HospitalPatient build() {
            return new HospitalPatient(this);
        }
    }
}

```

고객 정보 entity

고객 관리 리스트를 만들기 위한 entity 설정입니다.
환자 ID, 환자 이름, 주민등록번호, 주소, 비고, 첫 방문일을
설정했습니다.

Setter 대신 빌더(builder) 패턴을 사용해 가독성과 유연성
을 높이고 유지 보수를 쉽게 하였습니다.
또한, Service에서 Setter를 이용해 정보를 저장했을 때 일
어날 수 있는 실수를 방지하는 것도 목적 중 하나입니다.

Back-End (Entity)

```

@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class ClinicHistory {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "patientId", nullable = false)
    private HospitalPatient hospitalPatient;
    @Column(nullable = false)
    @Enumerated(value = EnumType.STRING)
    private MedicalItem medicalItem;
    @Column(nullable = false)
    private Boolean isInsurance;
    @Column(nullable = false)
    private Double patientPrice;
    @Column(nullable = false)
    private LocalDate dateClinic;
    @Column(nullable = false)
    private LocalTime timeClinic;
    @Column(nullable = false)
    private Boolean isCalculate;

    public void putCalculate() {
        this.isCalculate = true;
    }

    private ClinicHistory(ClinicHistoryBuilder builder) {
        this.hospitalPatient = builder.hospitalPatient;
        this.medicalItem = builder.medicalItem;
        this.isInsurance = builder.isInsurance;
        this.patientPrice = builder.patientPrice;
        this.dateClinic = builder.dateClinic;
        this.timeClinic = builder.timeClinic;
        this.isCalculate = builder.isCalculate;
    }

    public static class ClinicHistoryBuilder implements
        CommonModelBuilder<ClinicHistory> {

        private final HospitalPatient hospitalPatient;
        private final MedicalItem medicalItem;
        private final Boolean isInsurance;
        private final Double patientPrice;
        private final LocalDate dateClinic;
        private final LocalTime timeClinic;
        private final Boolean isCalculate;

        public ClinicHistoryBuilder(HospitalPatient hospitalPatient, HistoryRequest
            request) {
            this.hospitalPatient = hospitalPatient;
            this.medicalItem = request.getMedicalItem();
            this.isInsurance = request.getIsInsurance();
            this.patientPrice = request.getIsInsurance() ?
                request.getMedicalItem().getInsurancePrice() :
                request.getMedicalItem().getNonInsurancePrice();
            this.dateClinic = LocalDate.now();
            this.timeClinic = LocalTime.now();
            this.isCalculate = false;
        }

        @Override
        public ClinicHistory build() {
            return new ClinicHistory(this);
        }
    }
}

```

진료 기록 entity

진료 기록 리스트를 만들기 위한 entity 설정입니다.
 진료ID, 급여 여부, 진료비, 진료일, 진료 시간, 정산 여부를
 설정했습니다.
 환자 ID를 FK로 가져오고 진료 내용은 enum으로 가져 왔
 습니다.

고객 관리 entity와 같이 Setter 대신 빌더(builder) 패턴을
 사용해 가독성과 유연성을 높이고 유지 보수를 쉽게 하였
 습니다.
 또한, Service에서 Setter를 이용해 정보를 저장했을 때 일
 어날 수 있는 실수를 방지하는 것도 염두에 두었습니다.

Back-End (Controller)

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/patient")
public class PatientController {
    private final PatientService patientService;

    @PostMapping("/data")
    public String setPatient(@RequestBody @Valid PatientRequest request) {
        patientService.setPatient(request);

        return "OK";
    }

    @GetMapping("/all")
    public List<PatientItem> getPatient() {
        List<PatientItem> result = patientService.getPatient();

        return result;
    }

    @PutMapping("/update-info/id/{id}")
    public String putPatientInfo(@PathVariable long id, @RequestBody @Valid
PatientInfoUpdateRequest request) {
        patientService.putPatientInfo(id, request);

        return "OK";
    }
}
```

고객 관리 Controller

API의 안내 데스크 역할을 하는 Controller입니다.
고객 정보를 등록하기 위해 고객에게 request 값을 받아 정보를 Service에게 넘겨주는 Create 기능. /Valid를 이용해 model에 조건을 달아놓음으로서 유효성을 검증하게 만들었습니다.

환자 정보를 가져오기 위해 리스트 형태의 model을 Service에게 넘겨주는 Read 기능.

환자 정보를 수정하기 위해 request 값을 받아 정보를 Service에게 넘겨주는 Update 기능을 넣었습니다.

Back-End (Controller)

```

@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/history")
public class HistoryController {
    private final HistoryService historyService;

    private final PatientService patientService;

    @PostMapping("/data/patient-id/{patientId}")
    public String setHistory(@PathVariable long patientId, @RequestBody @Valid HistoryRequest request)
    {
        HospitalPatient hospitalPatient = patientService.getData(patientId);
        historyService.setHistory(hospitalPatient, request);

        return "OK";
    }

    @GetMapping("/all/date/")
    public List<HistoryItem> getHistoriesByDate(@RequestParam("searchDate") @DateTimeFormat(pattern =
"yyyy-MM-dd")LocalDate searchDate) {
        return historyService.getHistoriesByDate(searchDate);
    }

    @GetMapping("/all/insurance")
    public List<HistoryInsuranceCorporationItem> getHistoriesByInsurance(@RequestParam("searchDate")
@DateTimeFormat(pattern = "yyyy-MM-dd")LocalDate searchDate) {
        return historyService.getHistoriesByInsurance(searchDate);
    }

    @GetMapping("/all/history-id/{historyId}")
    public HistoryResponse getHistory(@PathVariable long historyId) {
        return historyService.getHistory(historyId);
    }

    @PutMapping("/calculate/id/{id}")
    public String putCalculate(@PathVariable long id) {
        historyService.putCalculate(id);

        return "OK";
    }
}

```

진료 기록 Controller

API의 안내 데스크 역할을 하는 Controller입니다.

진료 기록을 등록하기 위해 request 값을 받고, FK인 고객 ID를 이용해 Patient-Service에 있는 getData로 PatientRepository에서 고객 정보를 가져와 History-Service로 넘겨주는 Create 기능. / Valid를 이용해 model에 조건을 달아놓음으로서 유효성을 검증하게 만들었습니다.

날짜를 입력하면 그에 맞는 기록 데이터를 가져오는 기능과 급여 항목이면서 결제가 완료된 상태의 기록 데이터를 가져오는 Read 기능.

고객 ID를 입력하면 고객의 데이터 기록과 진료 기록을 전부 가져오는 Read기능.

고객이 결제를 했을 때 결제 상태를 완료로 바꾸는 Update 기능을 넣었습니다.

Back-End (Service)

```

@Service
@RequiredArgsConstructor
public class PatientService {
    private final HospitalPatientRepository hospitalPatientRepository;

    public void setPatient(PatientRequest request) {
        HospitalPatient addData = new
        HospitalPatient.HospitalPatientBuilder(request).build();

        hospitalPatientRepository.save(addData);
    }

    public HospitalPatient getData(long id) {
        return hospitalPatientRepository.findById(id).orElseThrow();
    }

    public List<PatientItem> getPatient() {
        List<HospitalPatient> originList = hospitalPatientRepository.findAll();

        List<PatientItem> result = new LinkedList<>();
        originList.forEach(item -> result.add(new
        PatientItem.PatientItemBuilder(item).build()));

        return result;
    }

    public void putPatientInfo(long id, PatientInfoUpdateRequest request) {
        HospitalPatient originData =
        hospitalPatientRepository.findById(id).orElseThrow();
        originData.putInfoUpdate(request);

        hospitalPatientRepository.save(originData);
    }
}

```

고객 정보 Service

Patient-Controller에게 request값을 받아 빌더 패턴을 이용해 addData에 저장한 뒤 Repository에 저장하는 기능. / 요청 받은 고객 정보 저장

Repository에 저장된 데이터를 List형태로 Controller에게 넘겨주는 기능. / 모든 고객 정보를 List 형태로 불러온다.

고객의 Id와 request 값을 받아 빌더 패턴을 이용해 저장했던 정보를 수정하는 기능.

Back-End (Service)

```

@Service
@RequiredArgsConstructor
public class HistoryService {
    private final ClinicHistoryRepository clinicHistoryRepository;

    public void setHistory(HospitalPatient hospitalPatient, HistoryRequest request) {
        ClinicHistory addData = new ClinicHistory.ClinicHistoryBuilder(hospitalPatient,
            request).build();

        clinicHistoryRepository.save(addData);
    }

    public List<HistoryItem> getHistoriesByDate(LocalDate searchDate) {
        List<ClinicHistory> originList =
            clinicHistoryRepository.findAllByDateClinicOrderByIdDesc(searchDate);

        List<HistoryItem> result = new LinkedList<>();

        originList.forEach(item -> result.add(new
            HistoryItem.HistoryItemBuilder(item).build()));

        return result;
    }

    public List<HistoryInsuranceCorporationItem> getHistoriesByInsurance(LocalDate
        searchDate) {
        List<ClinicHistory> originList =
            clinicHistoryRepository.findAllByIsInsuranceAndDateClinicAndIsCalculateOrderByIdDesc(true,
                searchDate, true);

        List<HistoryInsuranceCorporationItem> result = new LinkedList<>();

        originList.forEach(item -> result.add(new
            HistoryInsuranceCorporationItem.HistoryInsuranceCorporationItemBuilder(item).build()));

        return result;
    }

    public HistoryResponse getHistory(long id) {
        ClinicHistory originData = clinicHistoryRepository.findById(id).orElseThrow();

        HistoryResponse result = new
            HistoryResponse.HistoryResponseBuilder(originData).build();

        return result;
    }

    public void putCalculate(long id) {
        ClinicHistory originData = clinicHistoryRepository.findById(id).orElseThrow();
        originData.putCalculate();

        clinicHistoryRepository.save(originData);
    }
}

```

진료 기록 Service

History-Controller에게 request값을 받아 빌더 패턴을 이용해 addData에 저장한 뒤 Repository에 저장하는 기능. / 진료 기록 저장

Repository에 저장된 데이터를 List형태로 Controller에게 넘겨주는 기능. / 날짜를 입력하면 그 날에 저장된 진료 기록이 List형태로 나오는 기능, 급여 여부가 true 값인 진료 기록이 List로 나오는 기능 그리고 한 고객의 기록만을 가져오는 기능이 있습니다.

고객의 Id를 입력하면 정산이 완료되는 수정 기능. / 기본값인 false를 true로 바꿔주는 기능.

Part 5

Front-End

YOUR
IDEAS
MATTER

Write them down :)

Front-End (메인 페이지)

```
<template>
  <div class="center">
    <h1>한의원 관리자</h1>
  </div>
  <div class="center">
    <el-button type="primary" size="large" :icon="UserFilled">고객 관리</el-
button>
    <el-button type="primary" size="large" :icon="Edit">고객 정보 수정</el-button>
    <el-button type="primary" size="large" :icon="Finished">결제</el-button>
    <el-button type="primary" size="large" :icon="EditPen">진료 기록 관리</el-
button>
  </div>
</template>
```

한의원 관리자

고객 관리

고객 정보 수정

결제

진료 기록 관리

한의원 관리자의 시작이 되는 메인 페이지입니다.

각 버튼을 누르면 해당하는 페이지로 이동하고 기능을 수행할 수 있습니다.

Front-End (고객 관리)

```
<template>
  <div>
    <div class="center">
      <h1>고객 관리</h1>
    </div>
    <div class="center">
      <el-button type="primary">신규 등록</el-button>
      <el-button type="info">뒤로 가기</el-button>
    </div>
    <el-table :data="tableData" style="width: 100%">
      <el-table-column fixed prop="date" label="ID" width="150"/>
      <el-table-column prop="name" label="이름" width="120"/>
      <el-table-column prop="state" label="주민등록번호" width="120"/>
      <el-table-column prop="city" label="휴대폰 번호" width="120"/>
      <el-table-column prop="address" label="주소" width="600"/>
      <el-table-column prop="zip" label="최근 방문일" width="120"/>
      <el-table-column fixed="right" label="비고" width="120">
        <template #default>
          <el-button link type="primary" size="small" @click="handleClick">상세보기</el-button>
          <el-button link type="primary" size="small">수정</el-button>
        </template>
      </el-table-column>
    </el-table>
  </div>
</template>
```

고객 관리					
		신규 등록		뒤로 가기	
ID	이름	주민등록번호	휴대폰 번호	주소	비고
					상세보기 수정
					상세보기 수정
					상세보기 수정
					상세보기 수정

고객 관리 페이지입니다.

Back-End에서 Read 기능에 해당하는 모든 고객 정보를 볼 수 있는 기능을 넣었고, 신규 등록 혹은 수정이 가능하도록 설계했습니다.

Front-End (고객 관리)

```
<template>
  <div class="center">
    <el-form :model="form" label-width="120px">
      <el-form-item label="이름">
        <el-input v-model="form.name"/>
      </el-form-item>
      <el-form-item label="주민등록번호">
        <el-input v-model="form.registration"/>
      </el-form-item>
      <el-form-item label="휴대폰 번호">
        <el-input v-model="form.phone"/>
      </el-form-item>
      <el-form-item label="주소">
        <el-input v-model="form.address"/>
      </el-form-item>
      <el-button :icon="Search">주소 검색</el-button>
      <el-form-item label="비고">
        <el-input v-model="form.desc" type="textarea"/>
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click="onSubmit">등록</el-button>
        <el-button>취소</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>
```

A screenshot of a web form for customer information registration. The form is titled '고객 정보 등록' (Customer Information Registration) and is centered on the page. It contains several input fields: '이름' (Name), '주민등록번호' (Residential Registration Number), '휴대폰 번호' (Mobile Phone Number), '주소' (Address), and '비고' (Remarks). The '주소' field has a search button labeled '주소 검색' (Address Search). At the bottom, there are two buttons: '등록' (Register) and '취소' (Cancel).

고객 정보 등록 페이지입니다.
Back-End에서 Create 기능에 해당하는 고객 정보를 등록할 수 있는 기능을 넣었습니다.

Front-End (고객 관리)

```

<template>
  <div class="demo-input-size">
    <el-input
      v-model="input1"
      class="w-50 m-2"
      size="large"
      placeholder="ID"
      :suffix-icon="Search"
    />
  </div>
  <div>
    <div> 이름 </div>
    <div class="demo-input-size">
      <el-input
        v-model="input1"
        class="w-50 m-2"
        size="large"
        placeholder="홍길동"
      />
    </div>
  </div>
  <div>
    <div> 휴대폰 번호 </div>
    <div class="demo-input-size">
      <el-input
        v-model="input1"
        class="w-50 m-2"
        size="large"
        placeholder="010-0000-0000"
      />
    </div>
  </div>
  <div>
    <div> 주소 </div>
    <div class="demo-input-size">
      <el-input
        v-model="input1"
        class="w-50 m-2"
        size="large"
        placeholder="주소를 입력해주세요"
      />
    </div>
    <div><el-button :icon="Search">주소 검색</el-button></div>
  </div>
  <div>
    <div> 비고 </div>
    <div><el-input
      v-model="textarea"
      :rows="2"
      type="textarea"
      placeholder="Please input"
    />
    </div>
  </div>
</template>

```

ID

이름

홍길동

휴대폰 번호

010-0000-0000

주소

주소를 입력해주세요

주소 검색

비고

Please input

고객 정보 수정 페이지입니다.
Back-End에서 Update 기능에 해당하는 고객 정보를 수정할 수 있는 기능을 넣었습니다.

Front-End (고객 관리)

```

<template>
  <div class="center">
    <div><h1>고객 등록 완료</h1></div>
    <div><strong>***님의 정보가 등록되었습니다.</strong></div>
    <el-button type="primary">돌아가기</el-button>
    <el-button type="success">추가 등록</el-button>
  </div>
</template>

```

```

<template>
  <div>
    <div><h1>고객 정보 수정 완료</h1></div>
    <div><strong>***님의 정보가 수정되었습니다.</strong></div>
    <el-button type="primary">돌아가기</el-button>
    <el-button type="success">추가 수정</el-button>
  </div>
</template>

```

고객 등록 완료

***님의 정보가 등록되었습니다.

[돌아가기](#)
[추가 등록](#)

고객 정보 수정 완료

***님의 정보가 수정되었습니다.

[돌아가기](#)
[추가 수정](#)

고객 등록 완료 페이지와 수정 완료 페이지입니다.
고객 등록을 완료한 후와 수정을 완료한 후에 뒤로 돌아가거나 추가 등록 및 수정을 할 수 있게 설계하였습니다.

Front-End (고객 관리)

```

<template>
  <div>
    <div class="demo-input-size">
      <el-input
        v-model="input1"
        class="w-50 m-2"
        size="large"
        placeholder="ID"
        :suffix-icon="Search"
      />
    </div>
    <div class="center"><el-button type="primary">결제 완료</el-button></div>
  </div>
</template>

```

```

<template>
  <div class="center">
    <div><h1>결제 되었습니다.</h1></div>
    <div><el-button type="primary">돌아가기</el-button></div>
  </div>
</template>

```

결제 완료

결제 되었습니다.

돌아가기

결제 페이지와 결제 완료 페이지입니다.

Front-End (진료 기록 관리)

```

<template>
  <div class="center">
    <el-button type="primary" :icon="Search">날짜별 검색</el-button>
    <el-button type="primary" :icon="Search">환자별 검색</el-button>
    <el-button type="primary">진료비 기록</el-button>
  </div>
  <div>
    <el-table
      :data="tableData"
      style="width: 100%; margin-bottom: 20px"
      row-key="id"
      border
      default-expand-all
    >
      <el-table-column prop="id" label="ID" sortable />
      <el-table-column prop="patientId" label="고객 ID" sortable />
      <el-table-column prop="medicalItem" label="진료 내용" sortable />
      <el-table-column prop="isInsurance" label="급여 여부" sortable />
      <el-table-column prop="patientPrice" label="진료비" sortable />
      <el-table-column prop="clinicDate" label="진료일" sortable />
      <el-table-column prop="clinicTime" label="진료 시간" sortable />
      <el-table-column prop="isCalculate" label="정산 여부" sortable />
    </el-table>
  </div>
</template>

```

ID	고객 ID	진료 내용	급여 여부	진료비	진료일	진료 시간	정산 여부
1	1						
2	2						
3	3						
4	4						

진료 기록 관리 페이지입니다.
Back-End에서 Read기능에 해당하는 모든 진료 기록 정보를 확인할 수 있습니다.

Front-End (진료 기록 관리)

```

<template>
  <div class="right">
    <el-button type="primary" :icon="ArrowLeft">이전 페이지</el-button>
  </div>
  <div>
    <el-table
      :data="tableData"
      style="width: 100%; margin-bottom: 20px"
      row-key="id"
      border
      default-expand-all
    >
      <el-table-column prop="id" label="ID" sortable />
      <el-table-column prop="patientName" label="이름" sortable />
      <el-table-column prop="patientPhone" label="휴대폰 번호" sortable />
      <el-table-column prop="registrationNumber" label="주민등록번호" sortable />
      <el-table-column prop="address" label="주소" sortable />
      <el-table-column prop="firstVisit" label="주민등록번호" sortable />
      <el-table-column prop="memo" label="비고" sortable />
      <el-table-column prop="historyId" label="진료 번호" sortable />
      <el-table-column prop="isInsurance" label="급여 여부" sortable />
      <el-table-column prop="medicalItem" label="진료 내용" sortable />
      <el-table-column prop="patientPrice" label="진료비" sortable />
      <el-table-column prop="clinicDate" label="진료일" sortable />
      <el-table-column prop="clinicTime" label="진료 시간" sortable />
      <el-table-column prop="isCalculate" label="청산 여부" sortable />
    </el-table>
  </div>
</template>

```

[< 이전 페이지](#)

ID	이름	휴대폰 번호	주민등록번호	주소	주민등록번호	비고	진료 번호	급여 여부	진료 내용	진료비	진료일
1											
2											
3											
4											

환자별 기록 관리 페이지입니다.
 Back-End에서 Read기능에 해당하는 환자 한 명의 전체 기록을 확인할 수 있습니다.

Front-End (진료 기록 관리)

```

<template>
  <div class="right">
    <el-button type="primary" :icon="ArrowLeft">이전 페이지</el-button>
  </div>
  <div>
    <el-table
      :data="tableData"
      style="width: 100%; margin-bottom: 20px"
      row-key="id"
      border
      default-expand-all
    >
      <el-table-column prop="id" label="ID" sortable />
      <el-table-column prop="patientName" label="이름" sortable />
      <el-table-column prop="patientPhone" label="휴대폰 번호" sortable />
      <el-table-column prop="registrationNumber" label="주민등록번호" sortable />
      <el-table-column prop="isInsurance" label="급여 여부" sortable />
      <el-table-column prop="medicalItem" label="진료 내용" sortable />
      <el-table-column prop="patientPrice" label="진료비" sortable />
      <el-table-column prop="clinicDate" label="진료일" sortable />
      <el-table-column prop="clinicTime" label="진료 시간" sortable />
      <el-table-column prop="isCalculate" label="정산 여부" sortable />
    </el-table>
  </div>
</template>

```

[< 이전 페이지](#)

ID	이름	휴대폰 번호	주민등록번호	급여 여부	진료 내용	진료비	진료일	진료 시간	정산 여부
1									
2									
3									
4									

날짜별 기록 관리 페이지입니다.
 Back-End에서 Read기능에 해당하는 날짜별 진료 기록을 확인할 수 있습니다.

Front-End (진료 기록 관리)

```

<template>
  <div class="right">
    <el-button type="primary" :icon="ArrowLeft">이전 페이지</el-button>
  </div>
  <div>
    <el-table
      :data="tableData"
      style="width: 100%; margin-bottom: 20px"
      row-key="id"
      border
      default-expand-all
    >
      <el-table-column prop="patientName" label="이름" sortable />
      <el-table-column prop="patientPhone" label="휴대폰 번호" sortable />
      <el-table-column prop="registrationNumber" label="주민등록번호" sortable />
      <el-table-column prop="medicalItem" label="진료 내용" sortable />
      <el-table-column prop="patientPrice" label="진료비" sortable />
      <el-table-column prop="nonInsurancePrice" label="비급여시 비용" sortable />
      <el-table-column prop="billingAmount" label="공단 청구비" sortable />
      <el-table-column prop="clinicDate" label="진료일" sortable />
    </el-table>
  </div>
</template>

```

[< 이전 페이지](#)

이름	휴대폰 번호	주민등록번호	진료 내용	진료비	비급여시 비용	공단 청구비	진료일

보험 처리 기록 관리 페이지입니다.
 Back-End에서 Read기능에 해당하는 날짜별 보
 험 처리된 진료를 불러와 공단에 청구할 비용을
 확인할 수 있습니다.

Front-End (진료 기록 관리)

```
<template>
  <div>
    <div class="demo-date-picker">
      <div class="block">
        <span class="demonstration">날짜</span>
        <el-date-picker
          v-model="value1"
          type="date"
          placeholder="Pick a date"
          :default-value="new Date(2023, 1, 1)"
        />
      </div>
    </div>
    <div class="center"><el-button type="primary">검색</el-button></div>
  </div>
</template>
```

```
<template>
  <div>
    <div class="demo-input-size">
      <el-input
        v-model="input1"
        class="w-50 m-2"
        size="large"
        placeholder="이름"
        :suffix-icon="Search"
      />
    </div>
    <div class="center"><el-button type="primary">검색</el-button></div>
  </div>
</template>
```

날짜

검색

이름

검색

날짜 검색 페이지와 환자 검색 페이지입니다.
 날짜별 진료 기록 혹은 보험 처리 기록에서 사용
 할 날짜를 검색합니다.
 특정 환자 진료 기록 페이지에서 사용할 환자의
 ID를 검색합니다.

Thank you for watching ☺
봐 주셔서 감사합니다