SKINNY CONTROLLERS IN RAILS

WHAT ABOUT FILTERING AND SORTING?

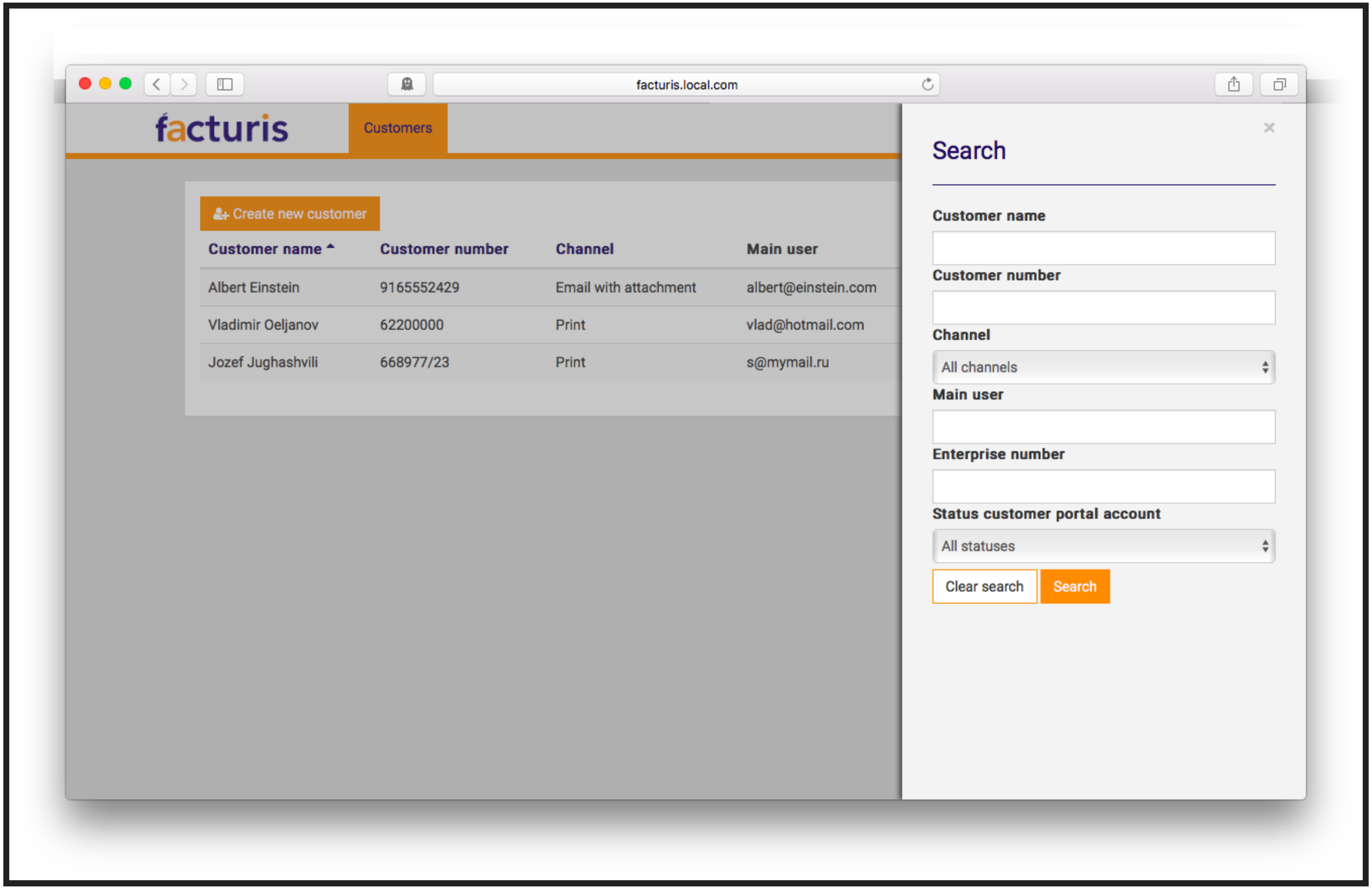
CONTROLLER ???

AUTHZ

HTTP ☞ APP ☞ HTTP

💬 DISCLAIMER

🔍 CASE STUDY


facturis.local.com

Customer name	Customer number	Channel	Main user
Albert Einstein	9165552429	Email with attachment	albert@einstein.com
Vladimir Oeljanov	62200000	Print	vlad@hotmail.com
Jozef Jughashvili	668977/23	Print	s@mymail.ru

Search

Customer name:
Customer number:
Channel: All channels
Main user:
Enterprise number:
Status customer portal account: All statuses

```ruby
class SubscriptionsController < ApplicationController
  load_and_authorize_resource
  check_authorization

  helper_method :sort_direction, :sort_column

  def index
    if search_params.empty?
      @subscriptions = @subscriptions.
        order("#{sort_column} #{sort_direction}").
        paginate(:page => params.fetch(:page, 1),
                 :per_page => params.fetch(:per_page, 10)).decorate
    else
      query, query_params = build_query
      @subscriptions = @subscriptions.
        where(query, query_params).
```

⚠️ ⚡ 💣 💀

FAILS THE S IN SOLID

REPETITION

HARD TO TEST

MISUNDERSTANDING OF ACTIVERECORD RELATIONS

HARDCODED SQL

VISIBILITY OF ACTIONS

MAGIC NUMBERS

DOUBLE AUTHORIZATION CHECKS

```
load_and_authorize_resource
check_authorization
```

SHARING OF THE HELPER METHODS

```
helper_method :sort_direction, :sort_column

private

def sort_column
  %w(name customer_number preferred_channel)
    .include?(params[:sort]) ? params[:sort] : 'name'
end

def sort_direction
  %w(asc desc).include?(params[:direction]) ? params[:direction] : 'a
end
```

😱

```ruby
def index
  if search_params.empty?
    @subscriptions = @subscriptions.
      order("#{sort_column} #{sort_direction}").
      paginate(:page => params.fetch(:page, 1),
               :per_page => params.fetch(:per_page, 10)).decorate
  else
    query, query_params = build_query
    @subscriptions = @subscriptions.
      where(query, query_params).
      order("#{sort_column} #{sort_direction}").
      paginate(:page => params.fetch(:page, 1),
               :per_page => params.fetch(:per_page, 10)).decorate
  end
end
```

```ruby
def search_params
  p = params.permit(:name, :customer_number, :preferred_channel,
                    :email, :unique_enterprise_number, :status)
  p.delete_if { |k, v| v.blank? && v != false }
  p
end
```

```ruby
def build_query
  q = []
  p = {}

  if !search_params[:name].blank?
    q << 'name ilike :name'
    p[:name] = "%#{search_params[:name]}%"
  end
  # ...
  if !search_params[:status].blank?
    q << 'status = :status'
    p[:status] = search_params[:status]
  end
  [q.join(' AND '), p]
end
```

MODEL

```ruby
class Subscription < ActiveRecord::Base
  # Insert domain logic here
end
```

ENTER SCOPES

`scope(name, scope_options = {}), public`

Adds a class method for retrieving and querying objects. A scope represents a narrowing of a database query […].

```ruby
scope :preferred_channel, ->(preferred_channel) {
  where(preferred_channel: preferred_channel)
}
```

```
scope :name, ->(name) {
  where('name ILIKE ?', "%#{name}%")
}
```

```ruby
scope :status, ->(status) {
  where(status: status)
}
```

• • •

```ruby
def for_params(params)
  results = self.where(nil)
  sort = params.delete(:sort)
  direction = params.delete(:direction)
  direction = 'ASC' unless direction == 'DESC'
  results = results.reorder("#{sort} #{direction}") if sort
  results = results.name(params[:name]) if params[:name].present?
  results = results.status(params[:status]) if params[:status].preser
  # repeat for all the filters
  results
end
```

```ruby
class Subscription < ActiveRecord::Base
  default_scope { for_params(sort: 'customer_name') }
  scope :customer_name, ->(customer_name) { ... }
  scope :customer_number, ->(customer_number) { ... }
  scope :preferred_channel, ->(preferred_channel) { ... }
  scope :email, ->(email) { ... }
  scope :unique_enterprise_number, ->(unique_enterprise_number) { ...
  scope :status, ->(status) { ... }
  def for_params(params)
    # ...
  end
end
```

```ruby
class SubscriptionsController < ApplicationController
  load_and_authorize_resource

  def index
    @subscriptions = @subscriptions.for_params(index_params)
  end

  private

  def index_params
    params.permit(:name, :customer_number,
                  :preferred_channel, :email, :status,
                  :sort, :direction, :page, :per_page)
  end
end
```

CONCLUSION

EMBRACE RAILS

DON'T BE HAPPY IF IT *JUST WORKS*

REFACTORING MAKES YOU HAPPY

THANK YOU