Goals

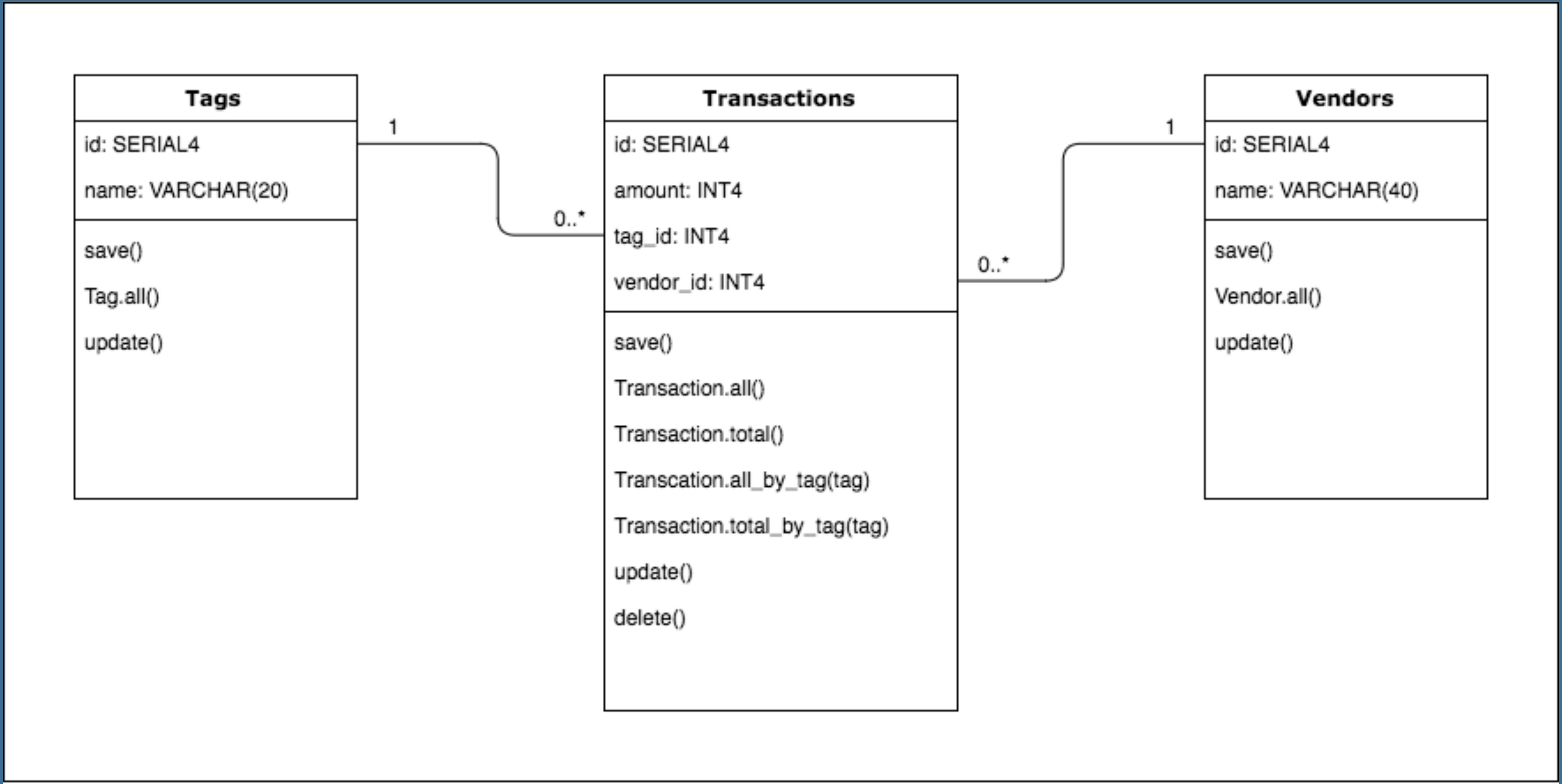
- As much functionality as possible but without adding so much that its unachievable
- Create an application that is complete and shippable
  - As few bugs as possible
- Simple, clean interface

Money Cashboard

Planning

- Mostly planned back end:
  - db tables
  - models/methods,
  - controllers/views
- Trello Board
- Class diagram
- Planned the front end very early on – created wireframe diagram last Friday

Planning – Class Diagram - Original



Tags

id: SERIAL4

name: VARCHAR(20)

save()

Tag.all()

update()

Transactions

id: SERIAL4

amount: INT4

tag_id: INT4

vendor_id: INT4

save()

Transaction.all()

Transaction.total()

Transcation.all_by_tag(tag)

Transaction.total_by_tag(tag)

update()

delete()

Vendors

id: SERIAL4

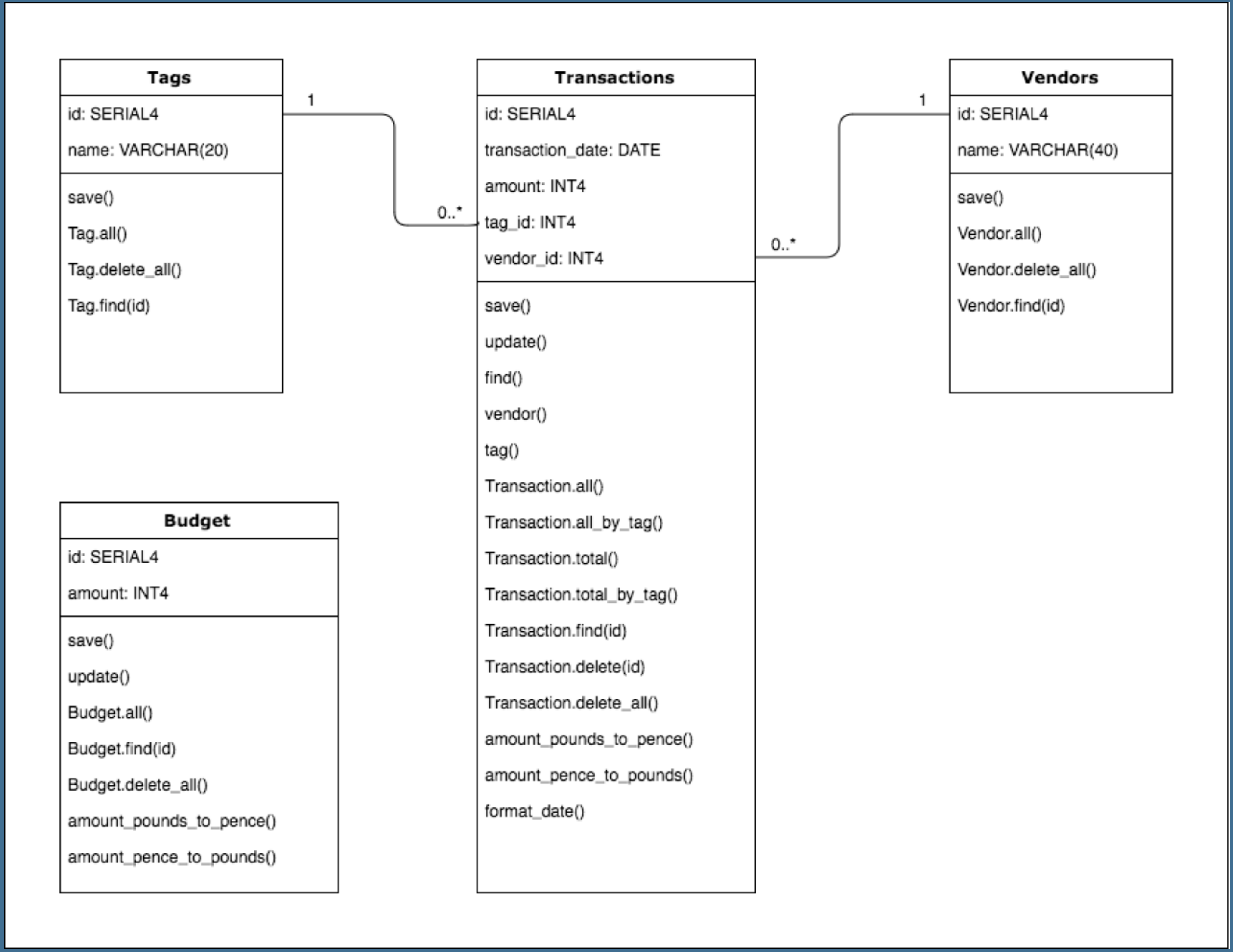
name: VARCHAR(40)

save()

Vendor.all()

update()

Planning – Class Diagram - Final



Tags

id: SERIAL4
name: VARCHAR(20)

save()
Tag.all()
Tag.delete_all()
Tag.find(id)

Transactions

id: SERIAL4
transaction_date: DATE
amount: INT4
tag_id: INT4
vendor_id: INT4

save()
update()
find()
vendor()
tag()
Transaction.all()
Transaction.all_by_tag()
Transaction.total()
Transaction.total_by_tag()
Transaction.find(id)
Transaction.delete(id)
Transaction.delete_all()
amount_pounds_to_pence()
amount_pence_to_pounds()
format_date()

Vendors

id: SERIAL4
name: VARCHAR(40)

save()
Vendor.all()
Vendor.delete_all()
Vendor.find(id)

Budget

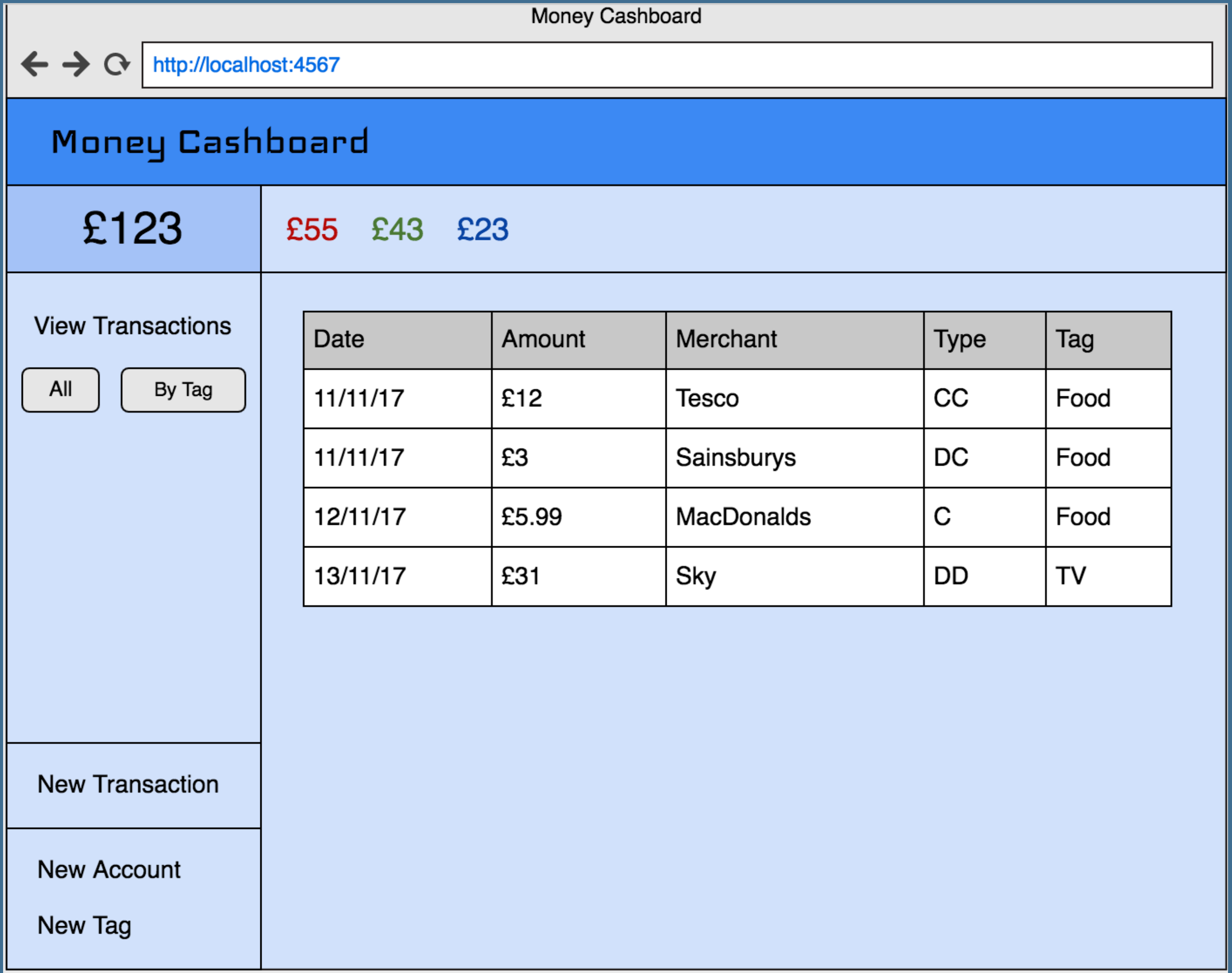
id: SERIAL4
amount: INT4

save()
update()
Budget.all()
Budget.find(id)
Budget.delete_all()
amount_pounds_to_pence()
amount_pence_to_pounds()

Planning – Front End

- Proto_persona.pptx
- User_requirements.xlsx
- User_journeys.xlsx

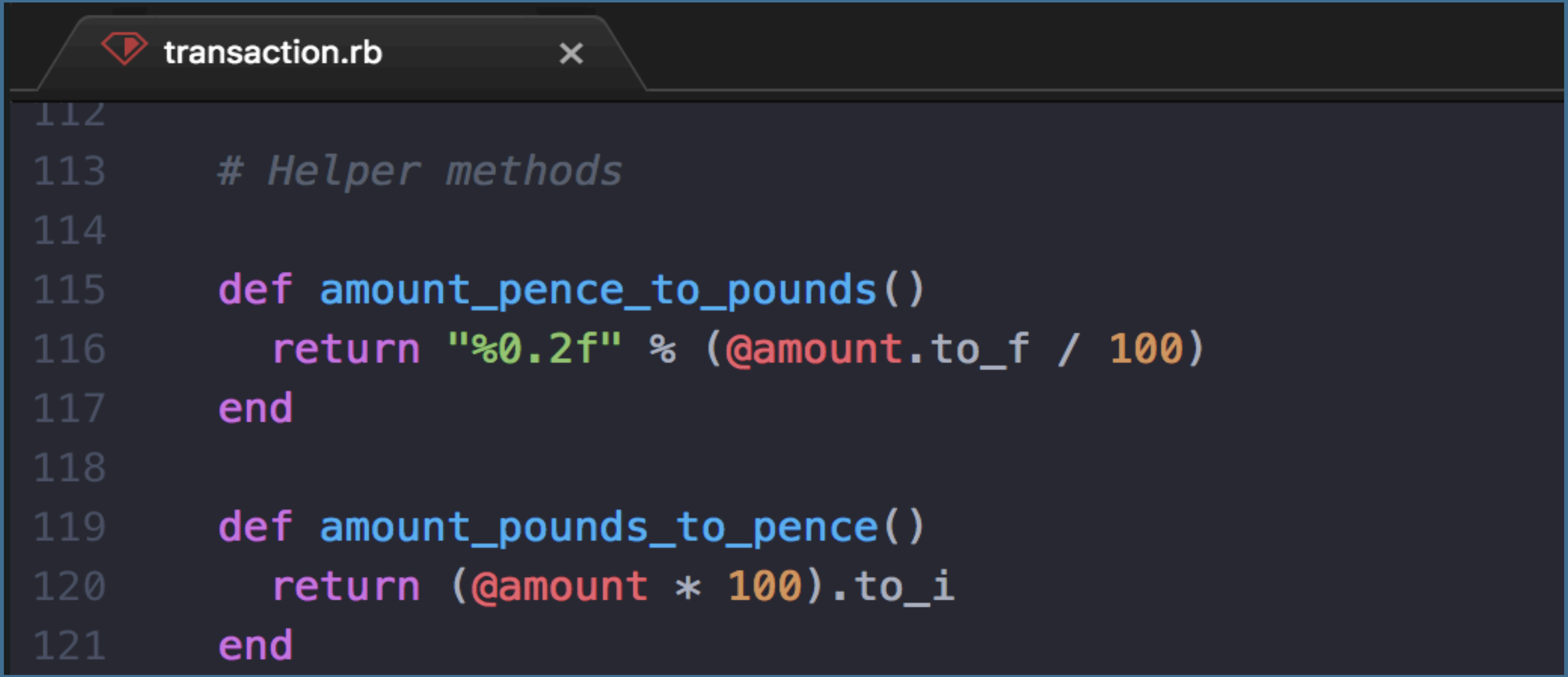
Planning – Wireframe



Date	Amount	Merchant	Type	Tag
11/11/17	£12	Tesco	CC	Food
11/11/17	£3	Sainsburys	DC	Food
12/11/17	£5.99	MacDonalds	C	Food
13/11/17	£31	Sky	DD	TV

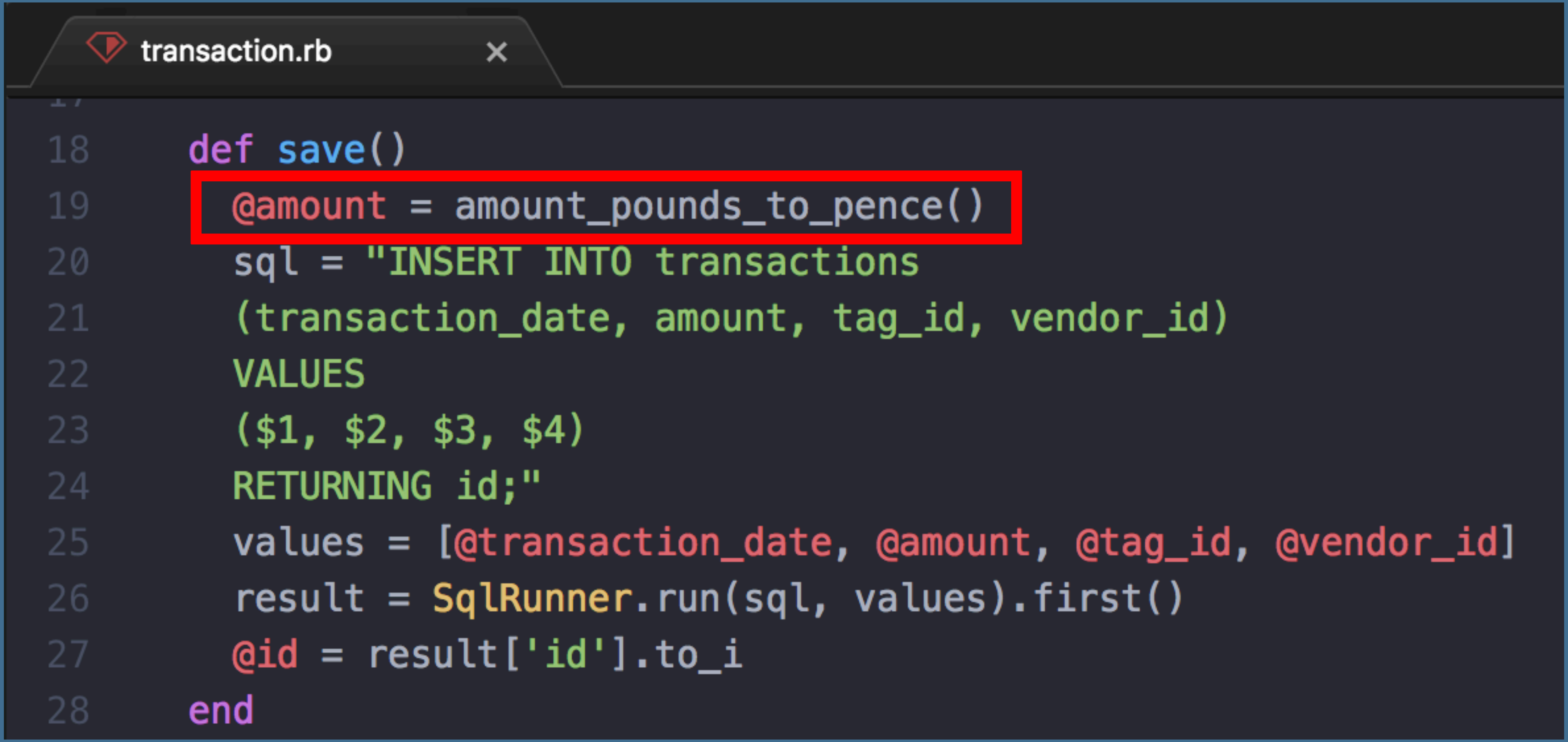
Demo

Explain Code – Amount Conversion - Methods



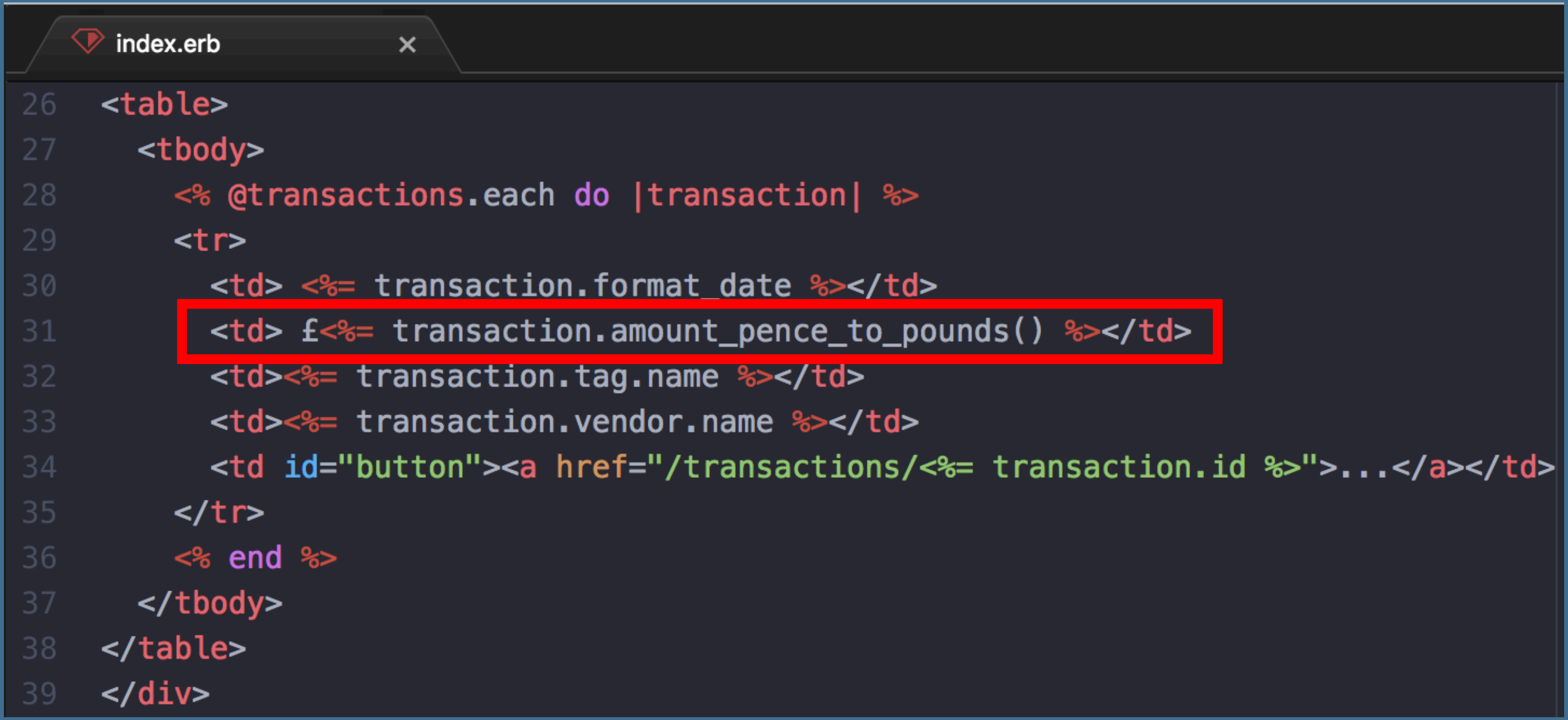
```ruby
  # Helper methods

  def amount_pence_to_pounds()
    return "%0.2f" % (@amount.to_f / 100)
  end

  def amount_pounds_to_pence()
    return (@amount * 100).to_i
  end
```

Explain Code – Amount Conversion - Save



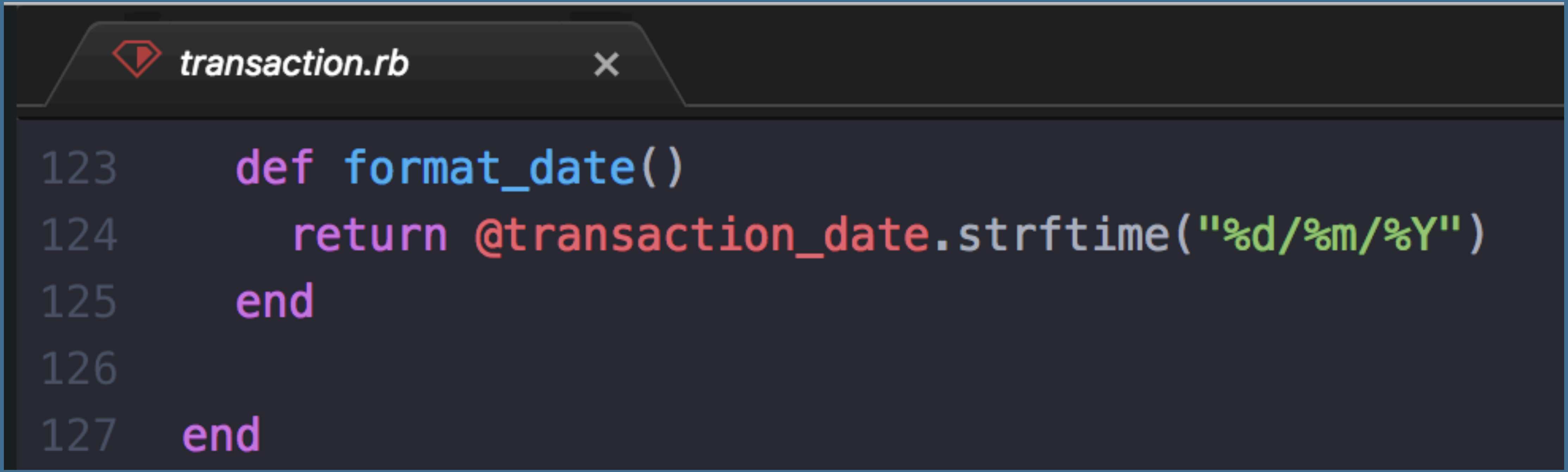
```ruby
  def save()
    @amount = amount_pounds_to_pence()
    sql = "INSERT INTO transactions
    (transaction_date, amount, tag_id, vendor_id)
    VALUES
    ($1, $2, $3, $4)
    RETURNING id;"
    values = [@transaction_date, @amount, @tag_id, @vendor_id]
    result = SqlRunner.run(sql, values).first()
    @id = result['id'].to_i
  end
```

Explain Code – Amount Conversion - View



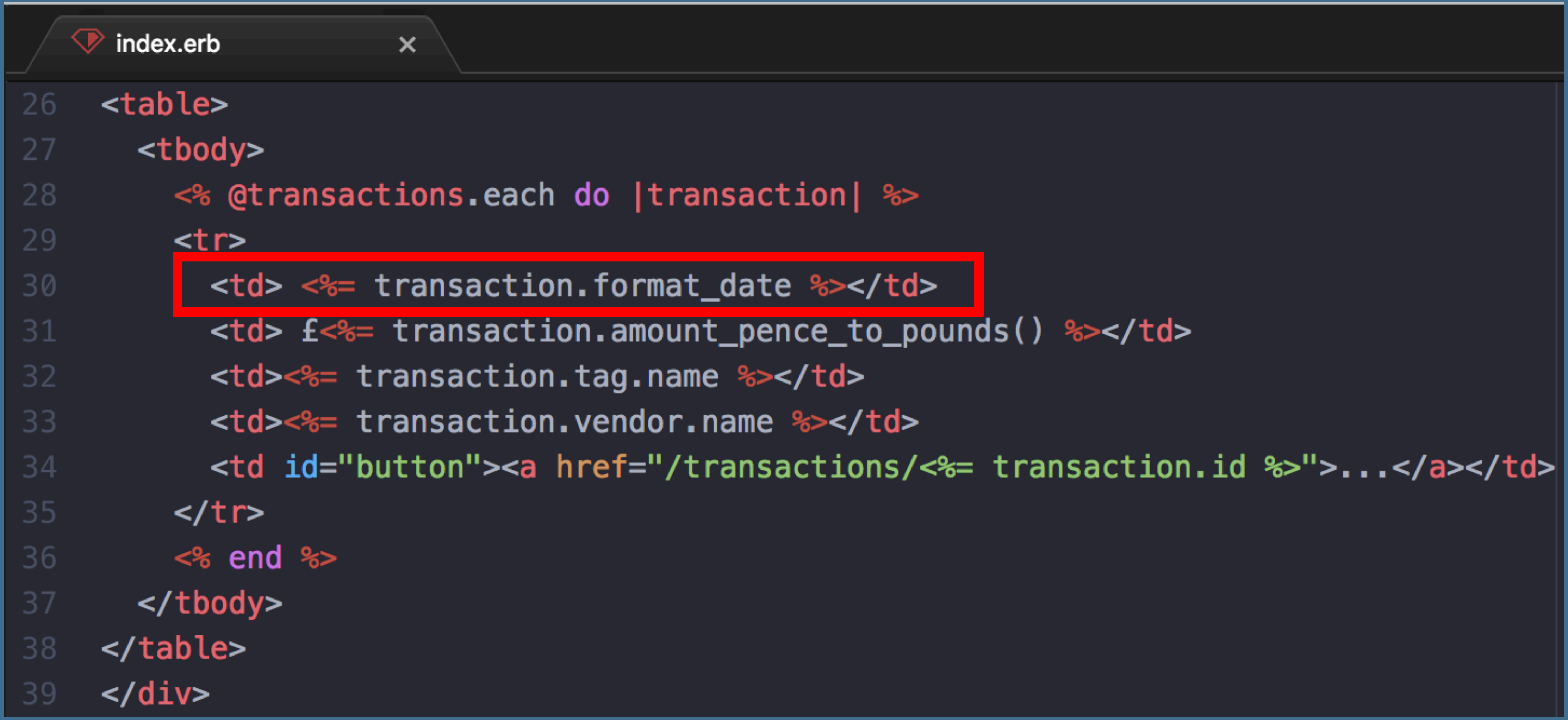
```
<table>
  <tbody>
    <% @transactions.each do |transaction| %>
    <tr>
      <td> <%= transaction.format_date %></td>
      <td> £<%= transaction.amount_pence_to_pounds() %></td>
      <td><%= transaction.tag.name %></td>
      <td><%= transaction.vendor.name %></td>
      <td id="button"><a href="/transactions/<%= transaction.id %>">...</a></td>
    </tr>
    <% end %>
  </tbody>
</table>
</div>
```

Explain Code – Date Conversion - Method



```ruby
  def format_date()
    return @transaction_date.strftime("%d/%m/%Y")
  end

end
```

Explain Code – Date Conversion - View



```
<table>
  <tbody>
    <% @transactions.each do |transaction| %>
    <tr>
      <td> <%= transaction.format_date %></td>
      <td> £<%= transaction.amount_pence_to_pounds() %></td>
      <td><%= transaction.tag.name %></td>
      <td><%= transaction.vendor.name %></td>
      <td id="button"><a href="/transactions/<%= transaction.id %>">...</a></td>
    </tr>
    <% end %>
  </tbody>
</table>
</div>
```

Improvements

- Helper methods (converting pounds/pence & pence/pounds) are duplicated in two models – need to refactor to move them into a separate helper class
- Need to improve how the website responds to resizing the browser
- Add multiple budgets
- Add accounts

Learning Points

- Planning is key
  - Planning the models, methods, controllers, views etc.
  - Having a clear vision of what the app should look like in the UI
- I need to get more outside my comfort zone in the next project (focus more on back end code)
- Sinatra is picky about ( ) in erb tags.
  - This works:

    ```
    <%= (Transaction.total.to_f() /100) %>
    ```

  - This doesn't!:

    ```
    <%= (Transaction.total.to_f /100) %>
    ```