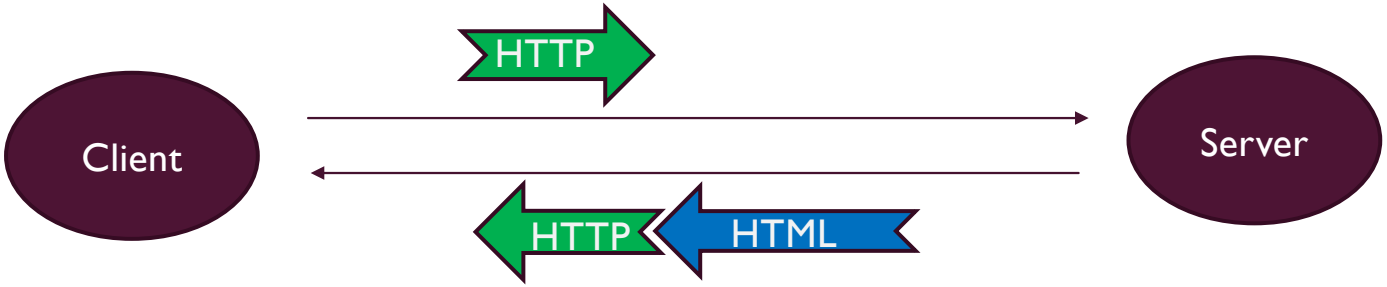
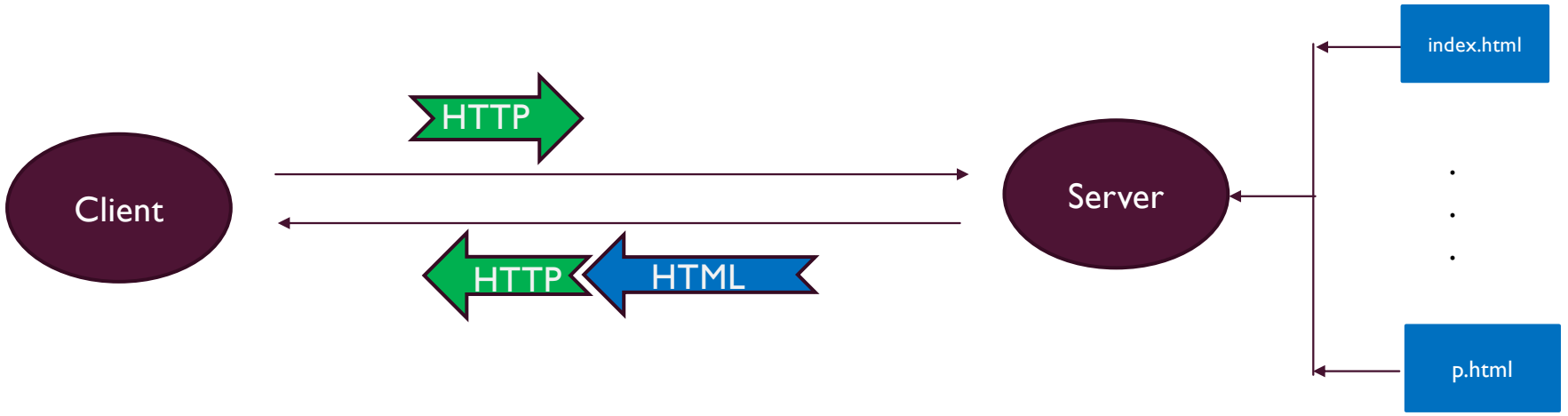
LABORATORIO APPLICAZIONE SW E SICUREZZA INFORMATICA

ROBERTO BERALDI

ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)
MODELLO DI INTERAZIONE

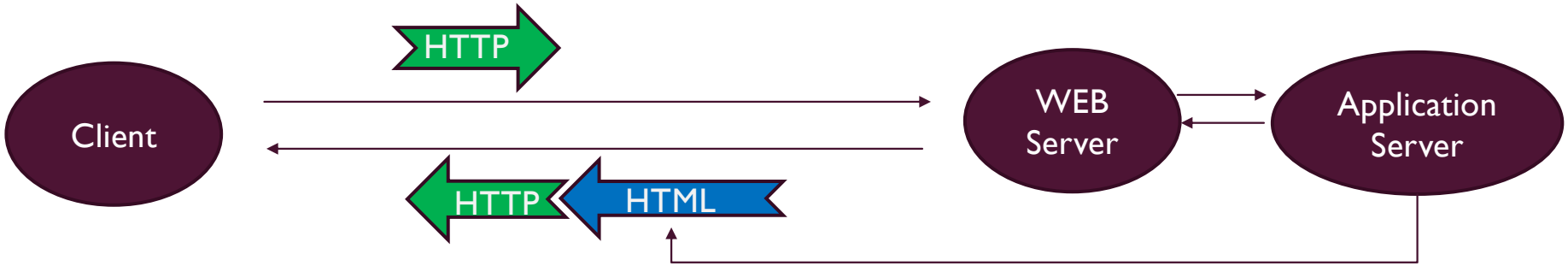


ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)
CONTENUTO STATICO



Static assets
Pagine pre-generate

ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)
GENERAZIONE DINAMICA DEI CONTENUTI



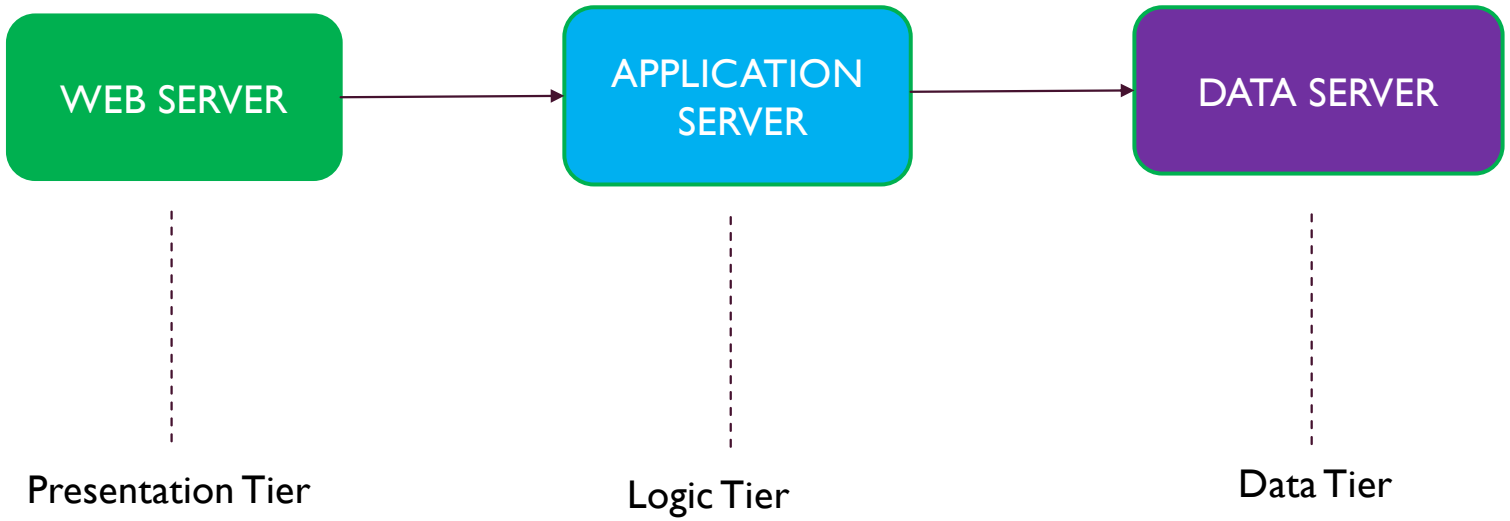
Input: dati dalla rachiesta
Output= pagina generate da un programma

ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)
GENERAZIONE DINAMICA DEI CONTENUTI

Input: dati dalla rachiesta
Output= pagina generate da un programma

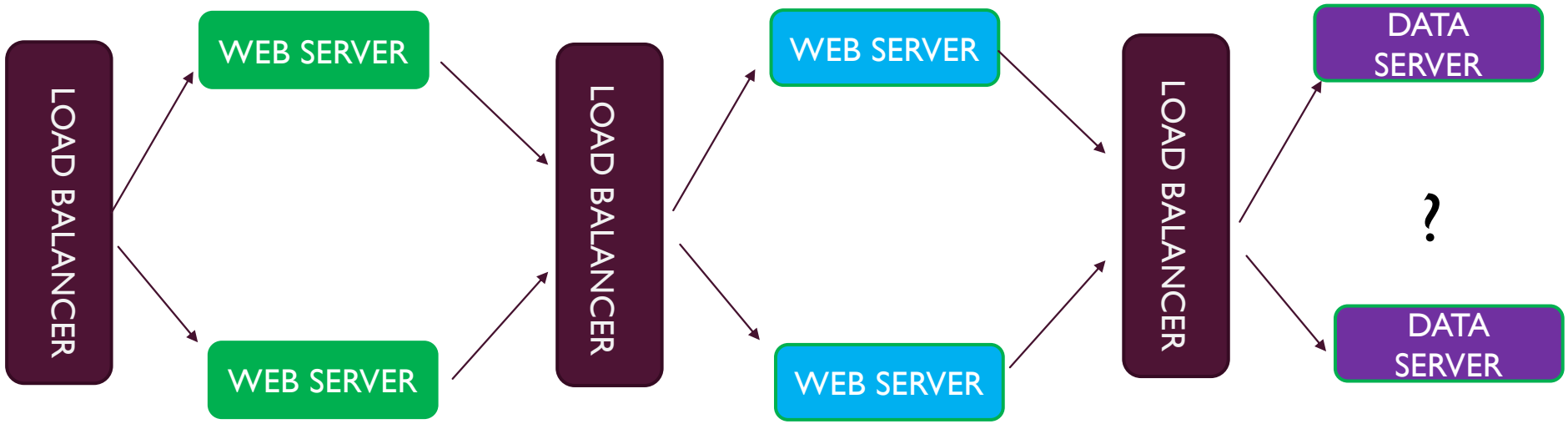
ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)

DIVISIONE IN LIVELLI



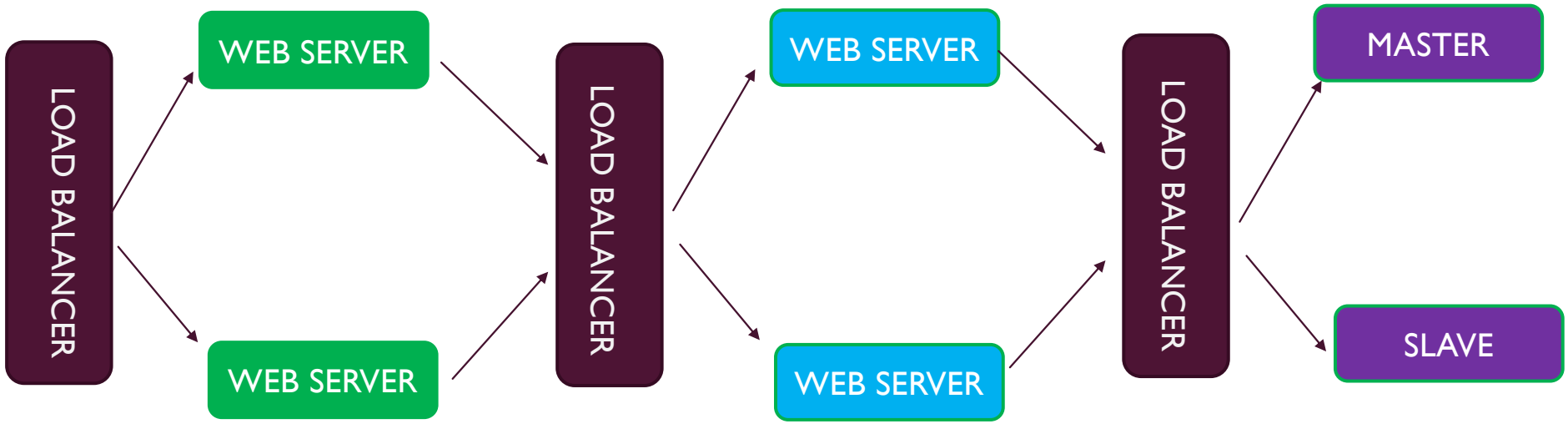
ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)

DIVISIONE IN LIVELLI



ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)

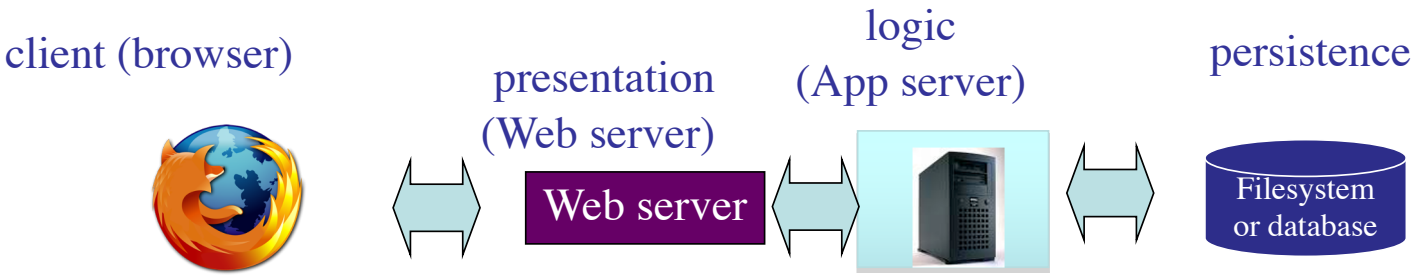
DIVISIONE IN LIVELLI



SHARE-NOTHING

MASTER-SLAVE

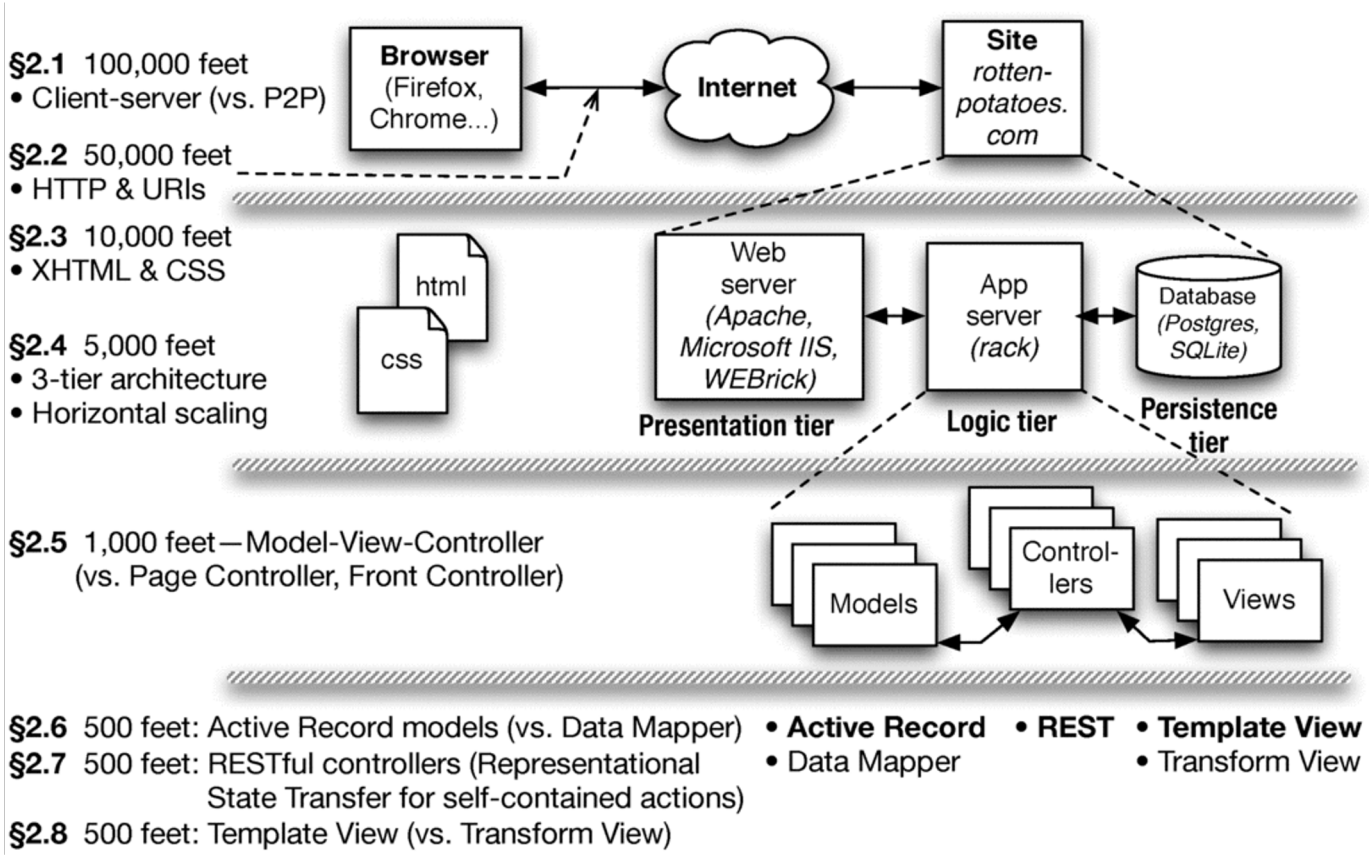
ARCHITETTURA 3-TIER E SCALING ORIZZONTALE (SEC. 2.4)
DIVISIONE IN LIVELLI



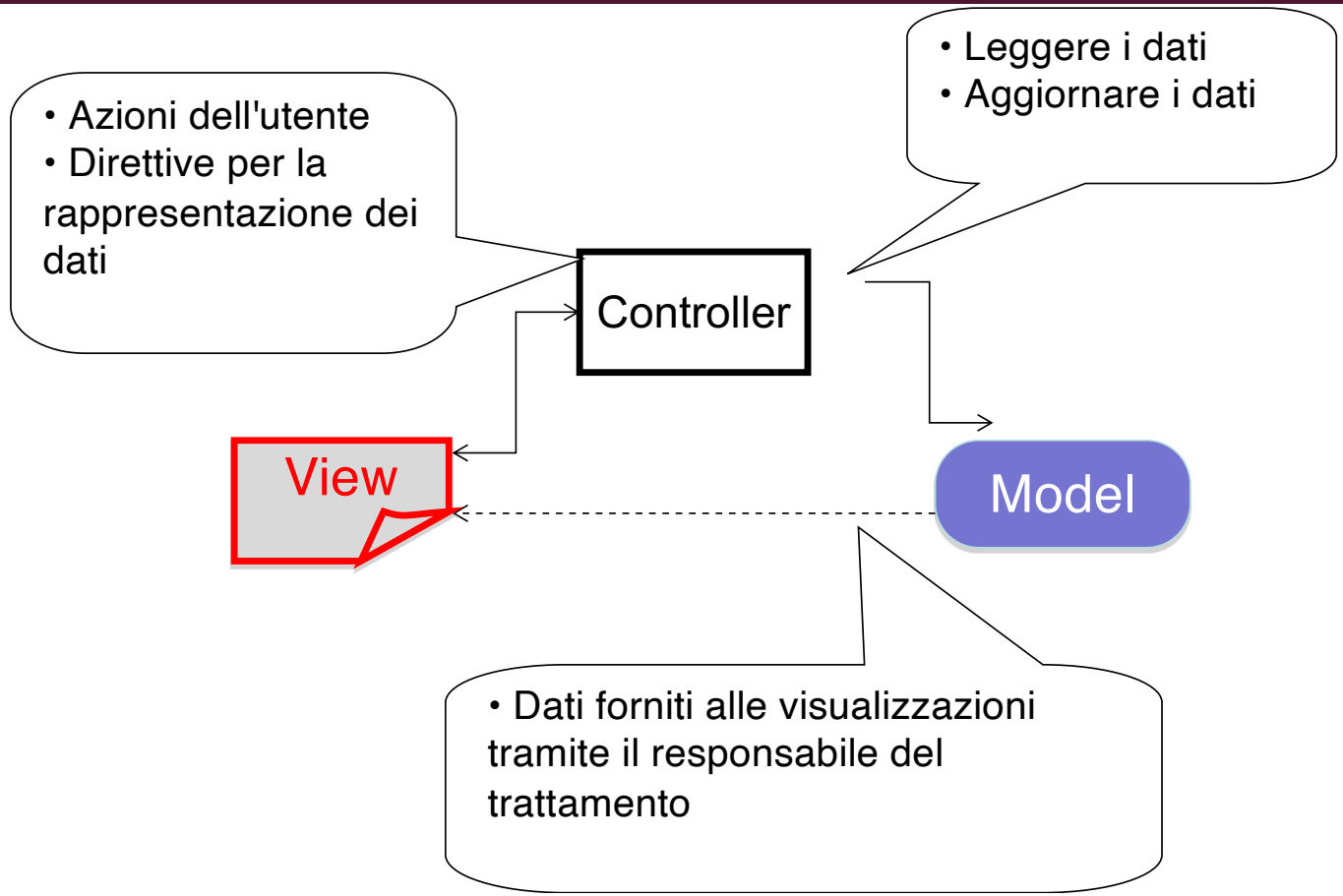
SUPPOERTO SVILUPPO APPS: IL RUOLO DEI FRAMEWORK

- Come fare a:
- URI "mappa" per correggere programma e funzione?
- passare argomenti?
- richiamare il programma sul server?
- gestire l'archiviazione persistente?
- gestire i cookie?
- gestire gli errori?
- l'output del pacchetto all'utente?
- I **framework** supportano queste attività comuni

MODEL-VEW-CONTROLLER (MVC)



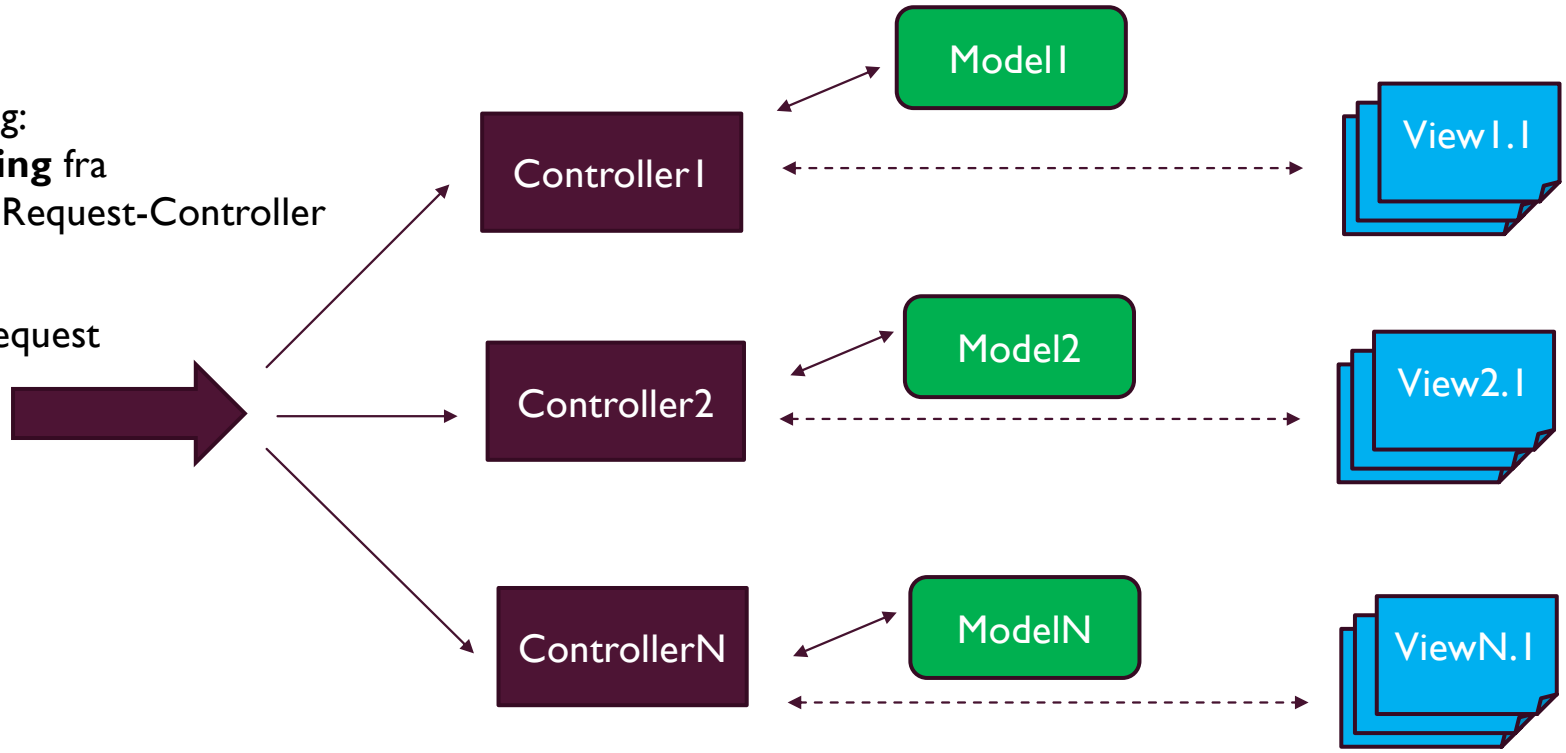
MVC


• Azioni dell'utente
• Direttive per la rappresentazione dei dati
• Leggere i dati
• Aggiornare i dati
Controller
View
Model
• Dati forniti alle visualizzazioni tramite il responsabile del trattamento

MVC (RAILS)

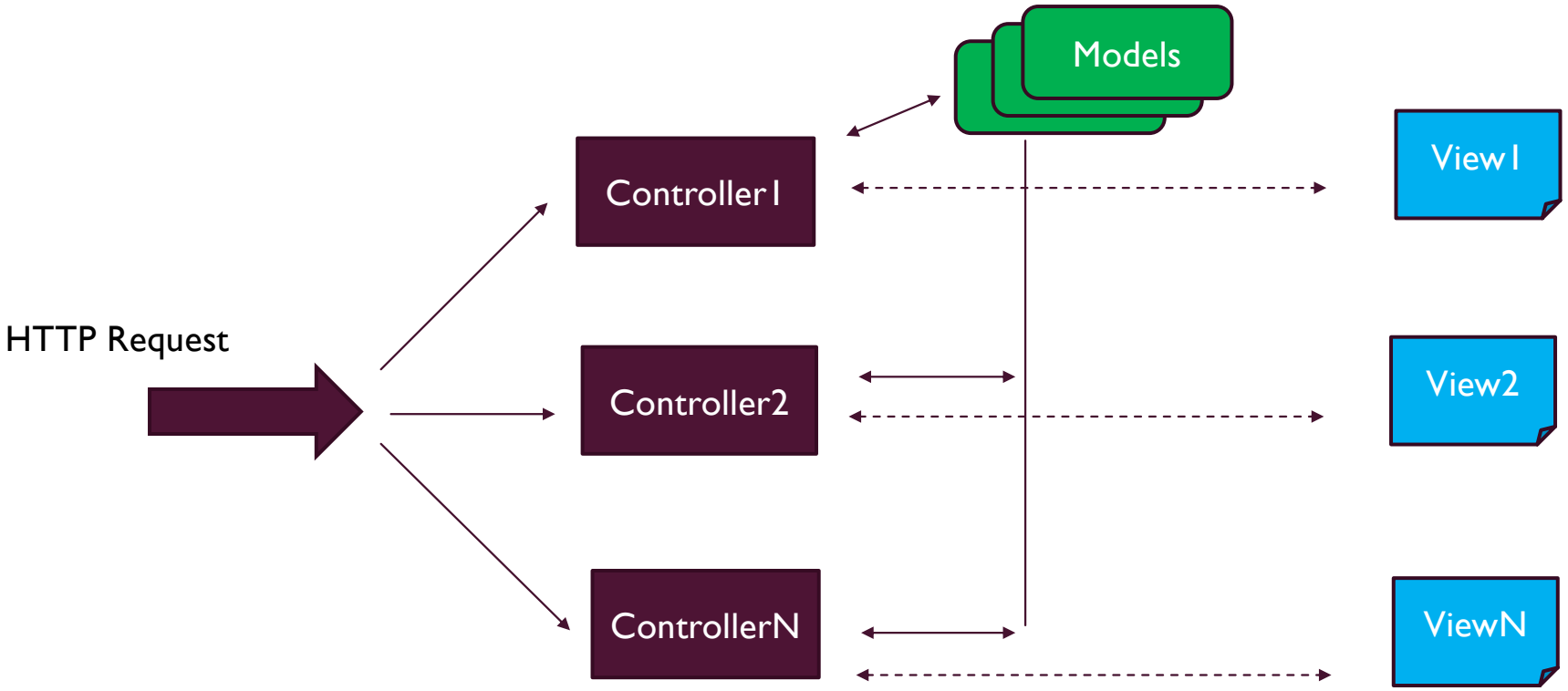
Routing:
Mapping fra
HTTP Request-Controller

HTTP Request

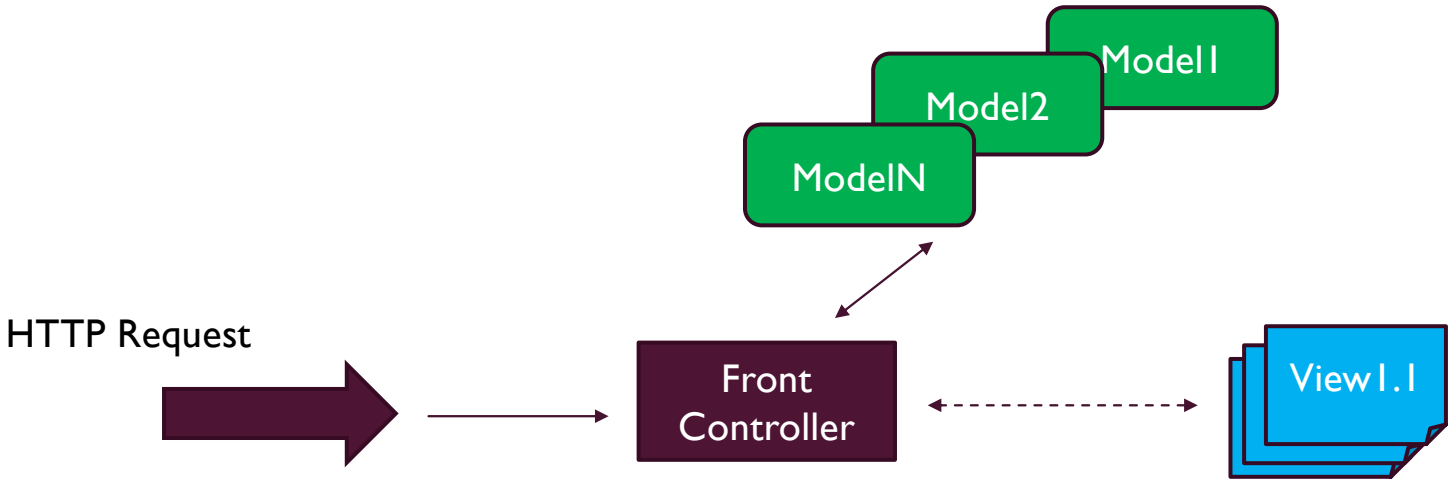


Controller-modello: 1-1, Controller-View 1-N

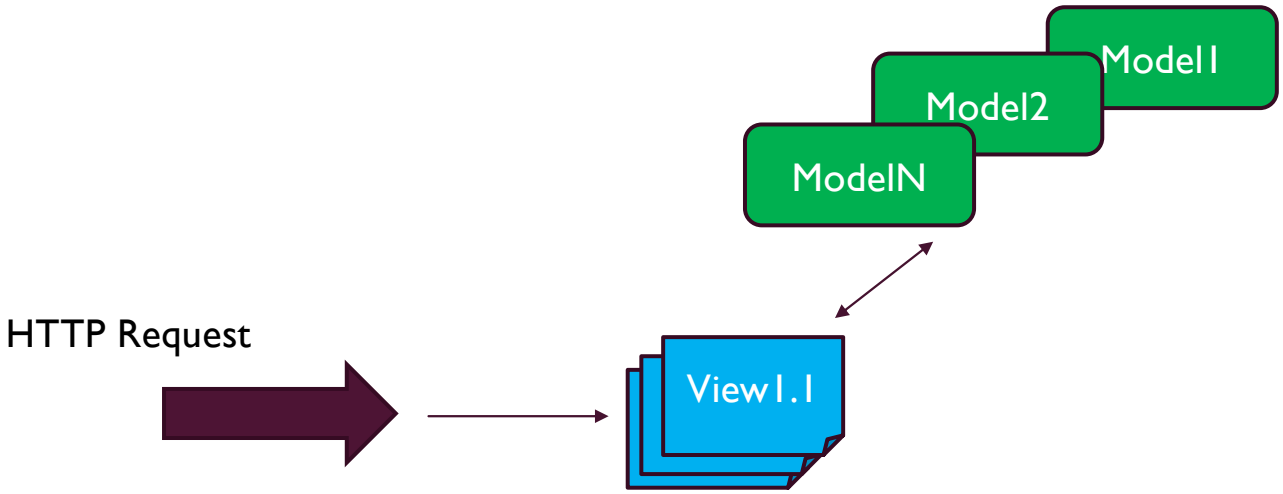
MVC (SINATRA)


Models
Controller1
View1
HTTP Request
Controller2
View2
ControllerN
ViewN

MVC (FRONT COMTROLLER, E.G. J2EE SERVLET)



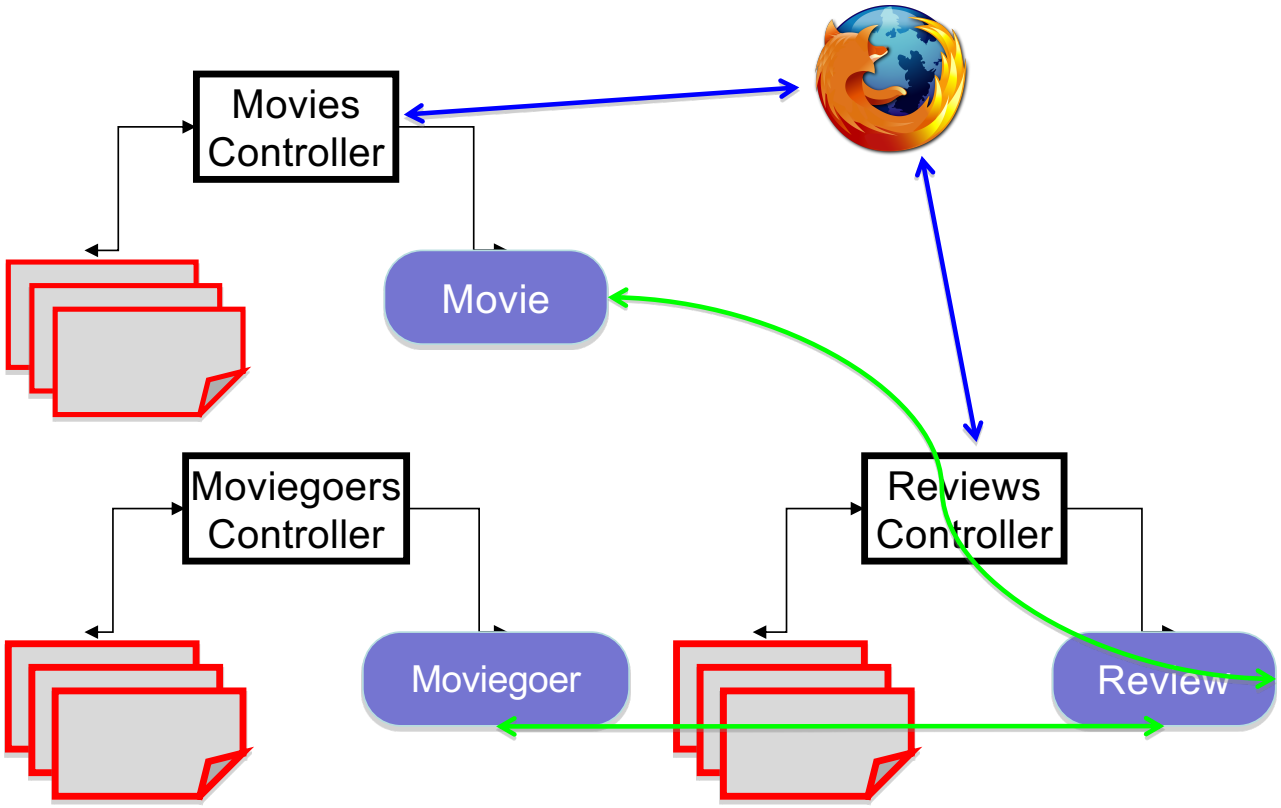
MVC (TEMPLATE VIEW, E.G. PHP)



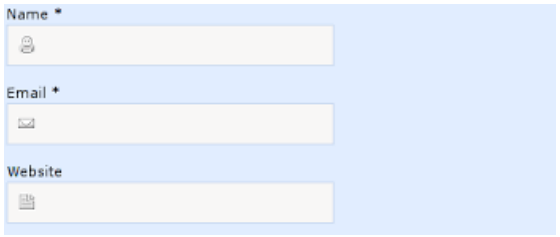
ESEMPIO ROR

```
%h1.pagename All Movies
%table#movies
  %thead
    %tr
      %th Movie Title
      %th Release Date
      %th More Info
  %tbody
    - @movies.each do |movie|
      %tr
        %td= movie.title
        %td= movie.release_date
        %td= link_to "More on #{movie.title}",
         movie_path(movie)
= link_to 'Add new movie', new_movie_path
```

ESEMPIO RAILS



INVIARE DATI AL SERVER (POST)



RISORSE E MODELLO REST

- Mediante il browser, è possible gestire non pagine web ma **RISORSE**
- Non solo 'leggere', ma anche create, aggiornare e cancellare
- Al browser il server invia una **rappresentazione** dello stato
- Il browser modifica la risorsa medinate un set finito di operazioni primitive

- **Representational State Transfer** (**RESTful**)

METODI CRUD

- Creare una Risorsa (**C**reate)
- Leggere una collezione di risorse (**R**ead)
- Leggere una sola risorsa (**R**ead)
- Aggiornare una risorsa (**U**pdate)
- Cancellare una risorsa (**D**elete)

RAILS ROUTIING (ESEMPIO)

Route	Action
`GET /movies/3`	Show info about movie whose ID=3
`POST /movies`	Create new movie from attached form data
`PUT /movies/5`	Update movie ID 5 from attached form data
`DELETE /movies/5`	Delete movie whose ID=5

ESEMPIO DI MAPPING CRUD-AZIONE (NEL CONTROLLER)

rake routes

```
I    GET /movies           {:action=>"index", :controller=>"movies"}
C   POST /movies           {:action=>"create", :controller=>"movies"}
     GET /movies/new       {:action=>"new", :controller=>"movies"}
     GET /movies/:id/edit {:action=>"edit", :controller=>"movies"}
R    GET /movies/:id       {:action=>"show", :controller=>"movies"}
U    PUT /movies/:id       {:action=>"update", :controller=>"movies"}
D DELETE /movies/:id       {:action=>"destroy", :controller=>"movies"}
```

ACTIVE RECORD PATTERN

- Collezione risorse = tabella data base relazionale
- Risorsa = una riga della tabella
- Operazioni CRUD su risorsa = Operazioni CRUD su tabella del DB
- Active Pattern associa ad una riga di una tabella un oggetto software