VOICES FROM THE TRIALS

'When I was a student I didn't worry much about time. But since I started working, time has become very precious … So saving time with this application is something I appreciate very much.'

'I'm a punctual person and I hate tempi morti: I tend to fill them all the time. Fluidtime is a good support for this, especially when I'm at work. I can look at my phone and, if I see a bus is arriving, I leave right away; otherwise, instead of waiting at the stop, I do something else before leaving.'

'It was very stressful for me to have to watch to see if a bus was coming; I prefer to look at the display…. I like it, it's nice and funny. I've learned to regulate my walking speed according to the information displayed on the watch-phone.'

'I can't be punctual; it's not in my nature and even Fluidtime could never help me…. What I like is the interface; it's really nice and entertaining, like a videogame. The best situation to use it is late at night, when there are few buses and waiting at the stop is very boring.'

The Fluidtime architecture

This documents describes the architecture of the Fluidtime system. Its design is based on the concepts described in [ARCH].

The architecture

Figure 1 describes the main component of the Fluidtime system:

- The red boxes on the left end-side represent the clients that connect through different channels (either in a push or pull form) to Fluidtime.
- The big white box in the middle represents the Fluidtime server itself, it is made by:
    - ***A component for each supported channel;***
    - ***A DB, that stores the user and application information;***
    - ***A rule engine, used to personalized the behavior of Fluitime, according with the time personality of the users;***
    - ***A scheduler, that drives the push channels;***
- The green boxes on the right end-side represent the content providers.

The Fluidtime system can be accessed through the following channels:

- ***Web PULL***, a web application is the default interface to interact with the fluidtime services;
- ***SMS PUSH and PULL***, a protocol of human readable messages has been defined to held the interaction;
- ***E-mail PUSH***, a protocol of human readable messages has been defined to held the interaction;
- ***HTTP PULL***, common HTTP GET are used to interact with fluidtime (require or subscribe services).

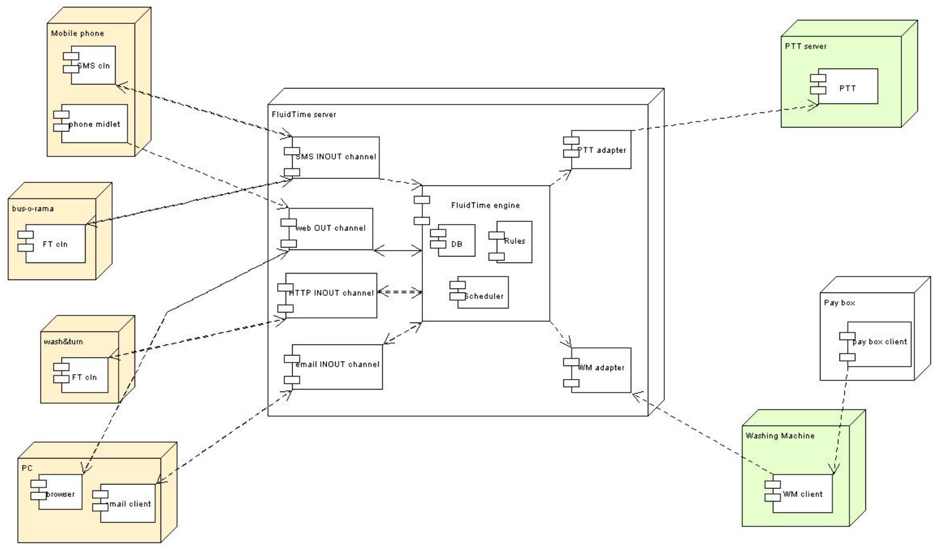

Figure 1: The main Fluidtime components.

The web channel is implemented by a series of web pages, while the other channels are java clients that connect through HTTP to the fluidtime server. In particular the system has the following clients:

- The ***email client***, which polls the fluidtime mail account, parses the incoming mail and translates them in HTTP calls to the server;
- The ***scheduler client***, which has a db of scheduled HTTP calls to the server, to have it send PUSH messages to the users;
- The ***sms client***, which polls the fluidtime GSM modem, parses the incoming SMSs and translates them in HTTP calls to the server;
- The ***washing machine client***, which is connected to the washing machine of the Laundry Service Ivrea and sends to the server the machine status updates.

Fluidtime can be accessed through the following devices:

- A mobile phone, through SMS push and pull or HTTP pull (via a midlet developed by the Fluidtime team);
- The busorama, through SMS push;
- The wash and turn, through SMS push;
- A common PC, through HTTP pull or email push.

Deep inside the architecture

Fluidtime architecture can be divided in tiers and layers (see Figure 2).

Layers

A ***layer*** contains components with similar concerns and allows to model the interaction between the application being developed and the technical platform (in our case java + the servlet engine) that hosts the application itself.
The lowest layer is the ***Infrastructure layer***, which provides the underlying technical and communications capabilities. Generally, the functionality in this layer is purchased from a middleware vendor; frequently, the purchased middleware will be customized.
The ***Services layer*** contains common utility functions that are useful in multiple tiers and by multiple classes of applications. Services include capabilities such as XML parsing and persistence.
The ***Application layer*** is where application and business functionality is implemented and where we actually apply the roles and responsibilities of four architectural tiers.

Tiers

A ***tier*** is a collection of software components, with similar concerns; it's a logical distribution unit, that can be installed on a single node. Different tiers can be installed on the same node.
The underlying layers take care of the transport between the different tiers.

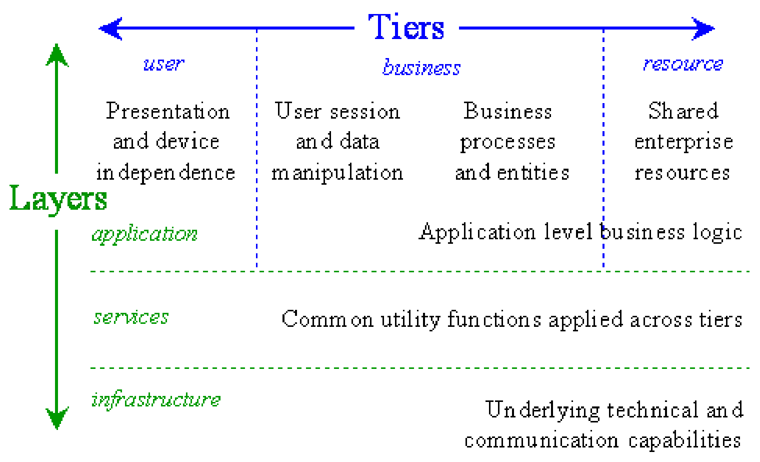
Tiers
user
business
resource
Presentation and device independence
User session and data manipulation
Business processes and entities
Shared enterprise resources
Layers
application
Application level business logic
services
Common utility functions applied across tiers
infrastructure
Underlying technical and communication capabilities

Figure 2: Layers and Tiers of the logical architecture.

The ***User Tier*** performs security authentication and authorization, establishes identity and does the initial message processing of the web pages or client calls.
The ***Business Tier*** is responsible for implementing business processes and entities and making their functions available via service-oriented interfaces. The Business Tier is responsible for providing business functionality and managing the integrity of enterprise resources. The business function in the Business Tier may itself be composed of finer-grain functions – in other words, it may be a business composition..
The ***Resource Tier*** is responsible for managing two basic types of resources; data and existing application services (information provider systems).